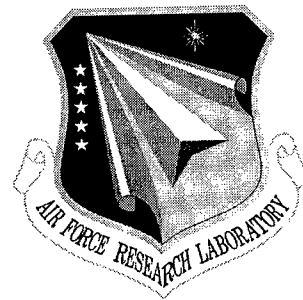


AFRL-IF-RS-TR-1998-38
Final Technical Report
April 1998



TEST BUS EVALUATION

Texas Instruments

Philip Dennis, Sue Vining, Wayne Daniel, and Jim Marischen

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

19980603 044

*Copyright 1993, Texas Instruments, Inc.
All Rights Reserved*

*This material may be reproduced by or for the U.S. Government pursuant to the copyright license
under clause at DFARS 252.227-7013 (October 1988).*

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

DTIC QUALITY INSPECTED 1

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-1998-38 has been reviewed and is approved for publication.

APPROVED:



FRANK H. BORN
Project Engineer

FOR THE DIRECTOR:



NORTHROP FOWLER, III, Technical Advisor
Information Technology Division
Information Directorate

If your address has changed or if you wish to be removed from the Air Force Research Laboratory Rome Research Site mailing list, or if the addressee is no longer employed by your organization, please notify AFRL/IFTB, 525 Brooks Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE April 1998		3. REPORT TYPE AND DATES COVERED Final Mar 92 - Jan 93
4. TITLE AND SUBTITLE TEST BUS EVALUATION			5. FUNDING NUMBERS C - F30602-88-D-0028/T56 PE - 62702F PR - 2338 TA - 01 WU - PI	
6. AUTHOR(S) Philip Dennis, Sue Vining, Wayne Daniel, and Jim Marischen				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Texas Instruments Defense Systems & Electronics Group 6500 Chase Oaks Blvd, Mail Stop 8407 Plano TX 75023			8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/IFTB 525 Brooks Road Rome NY 13441-4505			10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-1998-38	
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Frank H. Born/IFTB/(315) 330-4726				
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release;; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) <p>The purpose of the Test Bus Evaluation report was to evaluate and document the applications and impact of standard test buses on overall system testability. Current and proposed test bus architectures were surveyed and identified as most appropriate for coordinating testability approaches between the chip and the system level. Eleven test bus architectures were investigated.</p> <p>A Data Gathering task was first undertaken to characterize the major attributes of the test buses. These attributes include the bus architecture, current status, functions supported, interface and number of pins, protocol, intended uses and speed. This data collection included a library search of existing literature that is documented in the Bibliography.</p> <p>Current and planned test bus extensions were explored and documented in the report. Test Bus control issues were investigated and documented to include hardware controller applications to device and module test buses, and test bus control software. Standard test bus control languages and vector formats are also addressed in this report.</p>				
14. SUBJECT TERMS Testability, Built-in-Test, Bus Architecture, Test Software			15. NUMBER OF PAGES 232	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

TABLE OF CONTENTS.

EXECUTIVE SUMMARY.....	ix
PREFACE.....	xi
1. TEST BUS ARCHITECTURE SURVEY.....	1-1
1.1. Introduction.....	1-1
1.1.1. System Level Test Buses.....	1-1
1.1.2. Module Level Test Buses.....	1-2
1.1.3. Device Level Test Buses.....	1-4
1.2. IEEE 1149.1 / Joint Test Action Group (JTAG).....	1-5
1.2.1. Overview and Intended use.....	1-5
1.2.2. Current Status.....	1-6
1.2.3. Interface / Number of Pins.....	1-6
1.2.4. Architecture.....	1-7
1.2.5. Protocol.....	1-11
1.2.6. Instructions supported.....	1-12
1.2.7. Application.....	1-15
1.2.8. Bus timing.....	1-23
1.3. IEEE P1149.2 Extended Digital Serial Subset.....	1-23
1.3.1. Overview and Intended Use.....	1-23
1.3.2. Current Status.....	1-23
1.3.3. Interface / Number of Pins.....	1-23
1.3.4. Architecture.....	1-24
1.3.5. Functions Supported.....	1-27
1.4. IEEE P1149.4 Mixed-Signal Test Bus Standards.....	1-27
1.4.1. Overview and Intended use.....	1-27
1.4.2. Current Status.....	1-27
1.4.3. Architectural Elements of Proposed Frameworks.....	1-28
1.5. IEEE P1149.5/ TM-Bus.....	1-34
1.5.1. Overview and Intended Use.....	1-34
1.5.2. Current Status.....	1-35

1.5.3.	Interface/Number of Pins.....	1-36
1.5.4.	Architecture.....	1-37
1.5.5.	Bus Protocol.....	1-38
1.5.6.	Bus Timing.....	1-46
1.6.	IEEE 488 (GPIB).....	1-47
1.6.1.	Overview and Intended use.....	1-47
1.6.2.	Current Status.....	1-47
1.6.3.	Interface / Number of Pins.....	1-48
1.6.4.	Architecture.....	1-48
1.6.5.	Protocol.....	1-50
1.6.6.	Functions supported.....	1-51
1.6.7.	Speed.....	1-55
1.7.	MIL. STD. 1553B/ 1773.....	1-56
1.7.1.	Overview and Intended use.....	1-56
1.7.2.	Current Status.....	1-56
1.7.3.	Interface / Number of Pins.....	1-57
1.7.4.	Architecture.....	1-58
1.7.5.	Protocol.....	1-61
1.7.6.	Functions supported.....	1-62
1.7.7.	Speed.....	1-63
1.8.	High Speed Data Bus.....	1-65
1.8.1.	Overview and Intended use.....	1-65
1.8.2.	Current Status.....	1-65
1.8.3.	Interface / Number of Pins.....	1-65
1.8.4.	Architecture.....	1-66
1.8.5.	Protocol.....	1-67
1.8.6.	Functions supported.....	1-68
1.8.7.	Speed.....	1-69
1.9.	IEEE P1394.....	1-69
1.9.1.	Overview and Intended use.....	1-69
1.9.2.	Current Status.....	1-70
1.9.3.	Interface / Number of Pins.....	1-71

1.9.4. Architecture.....	1-71
1.9.5. Protocol.....	1-71
1.9.6. Functions supported.....	1-74
1.9.7. Speed.....	1-76
2. TEST BUS EXTENSIONS	2-1
2.1. IC.....	2-1
2.1.1. Internal Scan.....	2-1
2.1.2. BIST.....	2-6
2.1.3. Real-time event qualification.....	2-11
2.1.4. Embedded Software Emulation and Debug via IEEE 1149.1	2-19
2.1.5. Fault Emulation.....	2-24
2.2. Module.....	2-31
2.2.1. Intermodule testing.....	2-31
2.2.2. TM-Bus with TSMD.....	2-32
2.2.3. Extending IEEE 1149.1 in a Backplane Environment.....	2-35
2.3 System.....	2-42
3. TEST BUS CONTROL	3-1
3.1. Introduction.....	3-1
3.2. Test Bus Control Hardware.....	3-1
3.2.1. IEEE 1149.1.....	3-1
3.2.2. IEEE P1149.5 (TM-Bus).....	3-2
3.2.3. MIL-STD-1553B.....	3-2
3.3. Software Control.....	3-3
3.4. Supporting Control Languages.....	3-4
3.4.1. Boundary Scan Description Language (BSDL).....	3-4
3.4.2. Serial Vector Format (SVF).....	3-8
4. TEST BUS APPLICATIONS	4-1
4.1. Test Bus Architecture And Interfaces.....	4-1
4.1.1. Avionics System Architecture.....	4-1
4.1.2. System to Module.....	4-2
4.1.3. Module to IC.....	4-7
4.2. Specific Application Examples.....	4-8

4.2.1. F-16 Modular Mission Computer.	4-9
4.2.2. F-22 Vehicle Management System Test Bus Architecture.	4-12
4.2.3. F-22 Radar Array Power Supply.....	4-14
4.2.4. Aladdin Test Bus Architecture.....	4-17
4.2.5. Solid State Recorder (SSR).....	4-21
5. TEST BUS IMPACTS.....	5-1
5.1. IEEE 1149.1 (JTAG).....	5-1
5.1.1. Design.....	5-1
5.1.2. Manufacturing and Test (Fault Detection & Isolation).	5-7
5.1.3. Field Support and Maintenance.	5-9
5.1.4. Cost Summary.....	5-12
5.2. IEEE P1149.5/TM-Bus.	5-13
5.2.1. Design Considerations.....	5-13
5.2.2. Fault Detection/Isolation.	5-14
5.2.3. Cost Summary.....	5-14
5.3. System Buses.	5-15
6. TEST BUS EVALUATION RECOMMENDATIONS	6-1
6.1. Test Bus Design Criteria Recommendations.....	6-1
6.2. Management Recommendations.....	6-3
6.2.1 DoD Policies and Directives.....	6-3
6.2.2 DoD R&D Funding Recommendations.....	6-5
Appendix A. Commercially Available Test Bus Products.....	A-1
ASIC Foundries with 1149.1 Support.	A-1
FPGA Vendors with 1149.1 Support.	A-2
Appendix B. Bibliography.....	B-1
Appendix C. Acronyms.....	C-1

LIST OF FIGURES.

Figure 1.1-1.	A System Hierarchical Testing Approach.	1-1
Figure 1.1.1-1.	Test Functions Supported by System Buses	1-2
Figure 1.1.2-1.	TM-Bus Interfaces to Module and System Test Functions....	1-3
Figure 1.1.2-2.	Module Test Bus Interfaces to Ad-hoc BIT/BIST	1-4
Figure 1.1.3.	IEEE 1149.1 Interfaces to Device BIST	1-5
Figure 1.2.3-1.	IEEE 1149.1 Ring Connection.....	1-6
Figure 1.2.3-2.	IEEE 1149.1 Star Connection.....	1-7
Figure 1.2.4-1.	IEEE 1149.1 Architecture.....	1-8
Figure 1.2.4-2.	IEEE 1149.1 TDO Selection.....	1-9
Figure 1.2.4-3.	IEEE 1149.1 Generic Boundary Register Cell.....	1-10
Figure 1.2.5.	IEEE 1149.1 Test Access Port Protocol	1-12
Figure 1.2.7-1.	Pins-In and Pins-Out Testing using IEEE 1149.1	1-15
Figure 1.2.7-2.	Module Partitioning for Testability.....	1-16
Figure 1.2.7-3.	IC Partitioning For Testability.....	1-17
Figure 1.2.7-4.	Backplane Testing using IEEE 1149.1	1-18
Figure 1.2.7-5.	Testing Embedded Memory	1-19
Figure 1.2.7-6.	Testing Analog Logic	1-20
Figure 1.2.7-7.	PC Based Test Stations.....	1-20
Figure 1.2.7-8.	Board-Level Testing System using IEEE 1149.1	1-21
Figure 1.3.4-1.	Minimum P1149.2 Architecture.	1-24
Figure 1.3.4-2.	Logic Diagram for the SAP and Test Control Logic, Including the IDR and the Bypass Register.	1-26
Figure 1.4.3-1.	Parallel Connection of Analog Test Bus.	1-29
Figure 1.4.3-2.	Minimum Implementation of Switching Network	1-30
Figure 1.4.3-3.	Six-Switch Implementation of Switching Network	1-31
Figure 1.4.3-4.	Global BIT Resource as Defined in ICT Strawman.	1-33
Figure 1.5.3.	Architecture Interface	1-36
Figure 1.5.4.	P1149.5 Test Bus Architecture	1-37
Figure 1.5.5-1.	P1149.5 Protocol Layers	1-38
Figure 1.5.5-2.	Link-Layer Bus States	1-40

Figure 1.5.5-3.	Generic P1149.5 Message Format.....	1-43
Figure 1.6.4-1.	Basic 488 System Architecture	1-49
Figure 1.6.4-2.	Typical IEEE 488 System Configuration.....	1-50
Figure 1.6.6.	General IEEE 488 Functional Areas.....	1-54
Figure 1.7.1.	Typical Multiplex Data Bus Terminal Connectivity	
Figure 1.7.4-1.	Terminal Functional Elements	1-59
Figure 1.7.4-2.	Bus Level Topology	1-60
Figure 1.7.5.	1553/1773 Word Formats.....	1-62
Figure 1.8.4.	High Speed Data Bus Architecture.....	1-67
Figure 1.9.1.	P1394 Serial Bus Physical Topology.....	1-70
Figure 1.9.4.	Module Node Architecture	1-71
Figure 1.9.5.	IEEE P1394 Bus Protocol Element Connectivity	1-72
Figure 2.1.1-1.	Non-Scannable IC Logic Design	2-2
Figure 2.1.1-2.	Non-Scannable Module Circuit.....	2-2
Figure 2.1.1-3.	Scannable Modules B,C,D,E, and F	2-3
Figure 2.1.1-4.	Scannable Module A.....	2-4
Figure 2.1.1-5.	Scannable IC Logic Design	2-4
Figure 2.1.1-6.	Internal Scan in 1149.1 Architecture	2-5
Figure 2.1.2-1.	Embedded 1149.1 Test Controller	2-8
Figure 2.1.2-2.	Embedded 1149.1 Test Controller Architecture	2-9
Figure 2.1.2-3.	Generating Scan Offsets On-The-Fly.....	2-10
Figure 2.1.2-4.	Using Boundary Scan for PWB BIST	2-11
Figure 2.1.3-1.	Event Qualification Architecture	2-13
Figure 2.1.3-2.	EQUAL Protocol 1.....	2-15
Figure 2.1.3-3.	EQUAL Protocol 2.....	2-16
Figure 2.1.3-4.	EQUAL Protocol 3.....	2-17
Figure 2.1.3-5.	Global Event Qualification.....	2-18
Figure 2.1.4-1.	PC Based Test Facility.....	2-20
Figure 2.1.4-2.	Embedded Emulation Control	2-21
Figure 2.1.4-3.	Embedded Emulator User Interface Features	2-22
Figure 2.1.4-4.	Multi-Processor Embedded Emulation Control	2-23
Figure 2.1.4-5.	Using Boundary-Scan during System Debug.....	2-24

Figure 2.1.5-1.	Common Physical Fault Insertion Methods	2-26
Figure 2.1.5-2.	Fault Emulation Timing Diagram	2-28
Figure 2.1.5-3.	State Machine with Error Checking and Boundary Scannable Inputs	2-29
Figure 2.1.5-4.	Processor Based Design to Support Fault Emulation via Boundary Scan	2-30
Figure 2.2.1-1.	Module Interconnect Test Methods	2-31
Figure 2.2.2-1.	Fault Occurrence Correlation To Stress Data	2-33
Figure 2.2.2-2.	TSMD Architecture	2-34
Figure 2.2.3-1.	Board Using ASP Circuit	2-37
Figure 2.2.3-2.	Backplane ASP Connections	2-38
Figure 2.2.3-3.	Select and Acknowledge Protocols	2-39
Figure 2.2.3-4.	ASP Circuit Example	2-40
Figure 3.4.2.	Scan Chain of Six ICs	3-9
Figure 4.1.1.	An Advanced Avionics Architecture	4-2
Figure 4.1.2-1.	Subsystem Bus Architecture	4-4
Figure 4.1.2-2.	Module Test Bus Architecture	4-5
Figure 4.1.3.	Integrated Circuit Test Bus Architecture	4-8
Figure 4.2.1-1.	F-16 MMC Architecture	4-9
Figure 4.2.1-2.	F-16 MMC Test Bus Architecture	4-10
Figure 4.2.1-3.	F-16 MMC DP32 Module Test Architecture	4-11
Figure 4.2.2-1.	F-22 Vehicle Management System Architecture	4-12
Figure 4.2.3.	Array Power Supply Block Diagram	4-15
Figure 4.2.4-1	Basic Processing Module Architecture	4-18
Figure 4.2.4-2.	Aladdin Test Bus Architecture	4-19
Figure 4.2.4-3.	SEM-E Aladdin Test Bus Architecture.	4-20
Figure 4.2.5-1.	GigaBit Memory Unit	4-21
Figure 4.2.5-2.	Scan Path Architecture	4-22
Figure 5.1.1.	Standard Cell ASIC 1149.1 Overhead	5-3

LIST OF TABLES.

Table 1.4.3.	Function of Six-Switch Network.....	1-32
Table 1.5.5-1.	MTM -Bus Packet Types.....	1-42
Table 1.5.5-2.	MTM-Bus Command Codes.....	1-45
Table 1.6.3.	IEEE 488 Bus Signal Lines.....	1-48
Table 1.6.5-1.	IEEE 488.1 Messages.....	1-52
Table 1.6.5-2.	IEEE 488.2 Messages.....	1-53
Table 1.7.3.	1553B/1773 Data Bus/Coupling Requirements	1-58
Table 1.7.6.	Terminal Functional Processes	1-64
Table 1.8.3.	Bus Element Characteristics.....	1-66
Table 1.9.5.	Transaction Layer Transaction Types/Actions.....	1-73
Table 2.3-1.	System Bus Throughput Requirements	2-43
Table 2.3-2.	System Bus Expansion Capability	2-43
Table 3.4.1.	Boundary Scan Characteristics and Their Correlation to BSDL.....	3-6
Table 3.4.2.	Stable State Paths.....	3-12
Table 4.2.4.	Aladdin BPM Test Logic Summary.....	4-18
Table 5.1.1-1.	1149.1 IC Package/Pin Ratio	5-2
Table 5.1.1-2.	ASIC Cell Gate Count.....	5-3
Table 5.1.1-3.	Overhead Examples of Implementing IEEE 1149.1 and BIST.....	5-4
Table 5.1.3-1.	Feature Comparison for Standard Octal and Testability Octal Parts.....	5-10
Table 5.1.3-2.	Reliability Comparisons.....	5-11
Table 5.1.4.	Differences between Traditional (Ad-hoc) and Boundary- Scan Methods.....	5-12

EXECUTIVE SUMMARY

The purpose of the Test Bus Evaluation report was to evaluate and document the applications and impact of standard test buses on overall system testability. Current and proposed test bus architectures were surveyed and identified as most appropriate for coordinating testability approaches between the chip and the system level. The following test bus architectures were investigated;

- IEEE Std 1149.1, Standard Test Access Port and Boundary Scan Architecture
- IEEE Std P1149.2, Extended Digital Serial Test Bus
- IEEE Std P1149.4, Mixed-Signal Test Bus
- IEEE Std P1149.5, Standard Module Test and Maintenance Bus Protocol
- JIAWG TM-Bus (Joint Integrated Avionics Working Group)
- SAE TM-Bus (Society of Automotive Engineers)
- IEEE-488, General Purpose Interface Bus (GPIB)
- Mil-Std-1553B, Multiplexed Data Bus
- Mil-Std-1773, Fiber Optic Data Bus
- High Speed Data Bus
- IEEE P1394, High Speed Serial Bus

A Data Gathering task was first undertaken to characterize the major attributes of the test buses identified above. These attributes include the bus architecture, current status, functions supported, interface and number of pins, protocol, intended uses and speed. This data collection included a library search of existing literature that is documented in the Bibliography of this report.

Current and planned test bus extensions were explored and documented in the report. The following test and support techniques have been identified which work with the above test bus capabilities. Device level extensions include Built-In Self-Test (BIST), Internal Scan, Real-time Event Qualification, Embedded Design Debugger, and Fault Emulation. Module level extensions include Intermodule Testing, TM-Bus with Time Stress Measurement Device (TSMD), and Applying IEEE 1149.1 in a Backplane Environment. No system level extensions for test were identified, primarily due to the fact that test information is simply communicated via functional system buses.

Test bus control issues were investigated and documented to include hardware controller applications to device and module test buses, and test bus control software. The report addresses standard test bus control languages and vector formats.

A standard test and maintenance architecture is described, based on the JIAWG common modules, to identify how the different levels of test buses (device, module, system) interface and communicate in a typical avionics system. This architecture defines test bus data throughput and attributes required to support fault identification, fault location, false alarm filtering and fault logging.

The application of test buses in various defense contract system architectures was also studied and documented. This study included system architectures that were both existing and under development. The efforts chosen included the F-22 Vehicle Management System, F-22 Radar, F-16 Modular Mission Computer, the Aladdin Processor, and the Darpa Solid State Recorder. These systems included a mix of VHSIC, VLSI, multi-chip module, surface mount technology, and high density memory, along with FPGA, standard glue logic, and mixed signal technologies. The systems include both off-the-shelf parts, with limited testability, and custom parts, with extensive BIST.

The test bus architectures identified in the Survey were evaluated to determine their impacts on module and system level test. Advantages and disadvantages were identified and the 'value added' of each test bus architecture was defined. Design impacts, such as gate count, pins required, power, throughput, cost and reliability, are discussed in relationship to design verification, manufacturing test, and field support and maintenance.

Final recommendations were generated to support test bus architecture design, allocation and verification requirements. Recommendations addressed issues on system level architectures, proposed extensions, design automation tools and techniques. Management level recommendations have been made to the government on the level and extent that test bus standards and architectures should be specified in program requirements.

Finally, an Appendix is provided describing the current tools and vendor products available on the market today for the test bus standards.

PREFACE

The Test Bus Evaluation (TBE) report is based on years of test bus development and application's experience at Texas Instruments. Since the early 1980s TI has undertaken a key role in defining test buses; first, for the VHSIC program and then later as a key contributor in both JTAG (in Europe) and IEEE 1149.X (in the USA). TI is continuing its contribution to these efforts as a key player in the definition of the IEEE P1149.5 test bus.

A major portion of the TBE study has centered on the device level test bus due to the strong influence on system testability of the IEEE 1149.1 standard. Although module level test and maintenance buses have existed from the VHSIC program era, there has been little success in creating an industry standard module test bus until the emergence of IEEE P1149.5. The system buses identified in the TBE study are not dedicated test buses. They primarily communicate test status and pass down commands and data.

Special acknowledgement is given to Mr. Lee Whetsel and Mr. Wayne Daniel of TI's Semiconductor Group Test Technology Center and Mr. Greg Young, Don Sterba and Rex Sallade of TI's Defense Systems & Electronics Group Test Technology Laboratory for their contributions to this report.

system buses (MIL-STD-1553B, etc.) have always treated test and diagnostic data like any other mission data. It was not recognized until the late 1980s that dedicated fault management data at the system level was needed. Due to the complexity of the electronics and driven by stringent aircraft fault tolerance requirements, system level analysis in flight is needed to increase failure isolation and decrease pilot false alarms. The data for these analyses has put new demand on the old buses and new requirements on emerging system bus standards (such as the High Speed Data Bus, Mil-Std-1773 and IEEE P1394). Figure 1.1.1-1 depicts typical data and communication interfaces that an Aircraft Fault Management System has to address via these system buses.

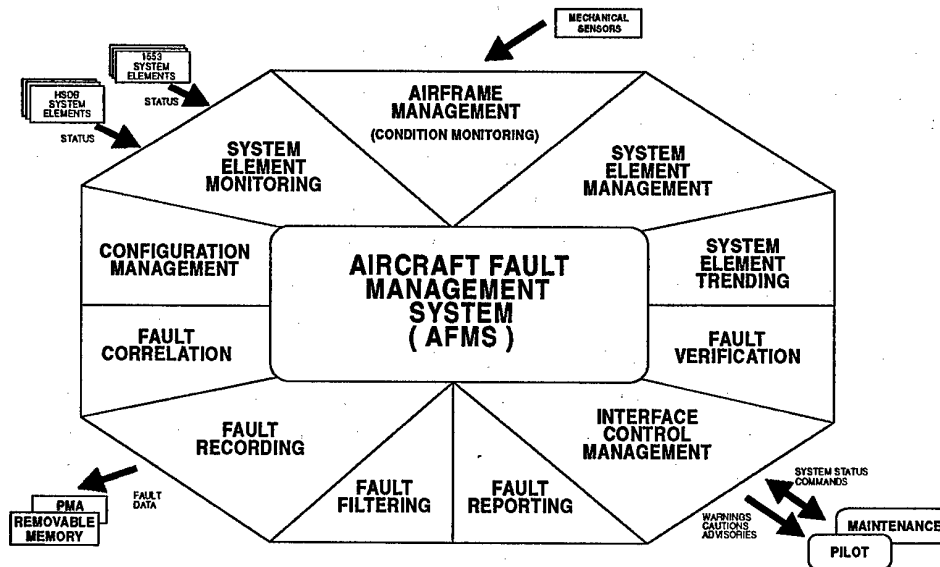


Figure 1.1.1-1. Test Functions Supported by System Buses

1.1.2. Module Level Test Buses.

During the VHSIC development it was realized that dedicated test buses would be required to support system to module to device test processes. The module level Test and Maintenance Bus (TM-Bus) and the device level Element Test and Maintenance Bus (ETM-Bus) were developed.

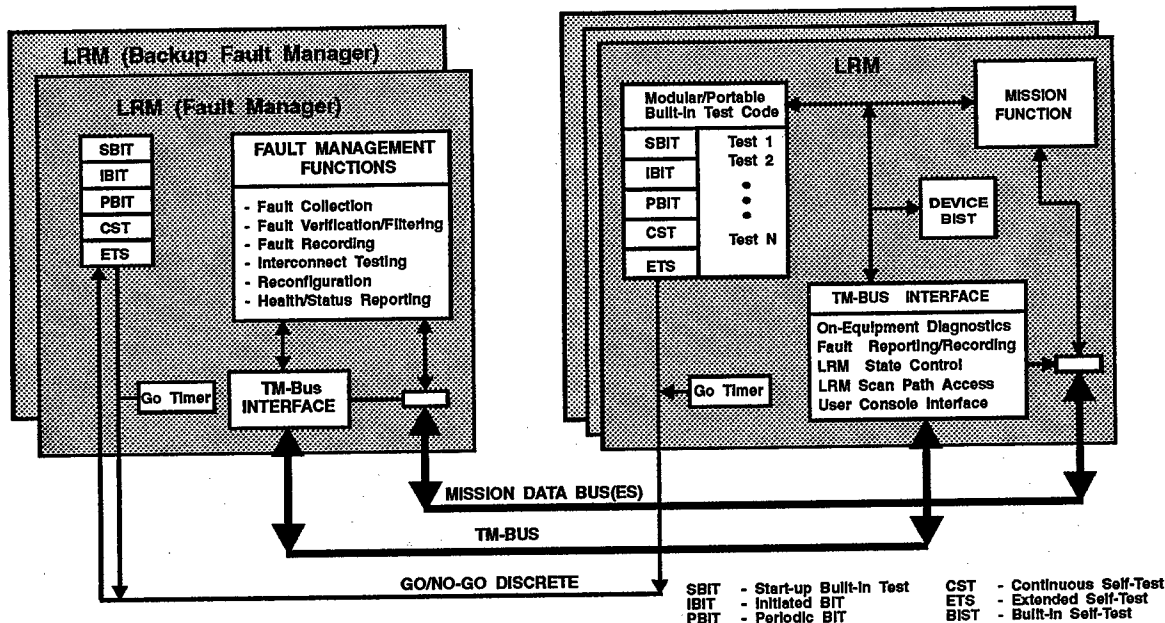


Figure 1.1.2-1. TM-Bus Interfaces to Module and System Test Functions

Module level buses for test were not new to the engineering world since the IEEE 488 General Purpose Interface Bus had its roots in interfacing modules to test equipment and instrumentation in the mid 1970s. But there had never been an effort, before VHSIC, to standardize an on-board data bus dedicated to test. The TM-Bus caught on, especially in mission and data processors, and was redefined many times until the most recent JIAWG common module effort. Figure 1.1.2-1 depicts this application and how the TM-Bus interfaces with other module and system test functions. But even after the JIAWG specification, implementations of the TM-Bus within defense contracts were different, preventing bus compatibility between common modules. An industry wide accepted solution would need to occur to solve this problem.

This solution has emerged out of the success of the IEEE 1149.1 device test bus efforts. Using lessons learned from this effort, defense (JIAWG) and commercial (SAE) groups have come together to create the IEEE P1149.5 Module Test and Maintenance (MTM) bus. Final balloting has assured a wide spread acceptance and expectation of an official IEEE 1149.5 standard in early 1993.

A key contribution of the IEEE P1149.5 will be the definition of the interface to the IEEE 1149.1 test bus. This is critical to achieve the ultimate hierarchical testability goals. Figure 1.1.1-2 depicts the module level test interfaces, including the interaction between the MTM bus, the IEEE 1149.1 device test bus and other ad-hoc BIT/BIST features.

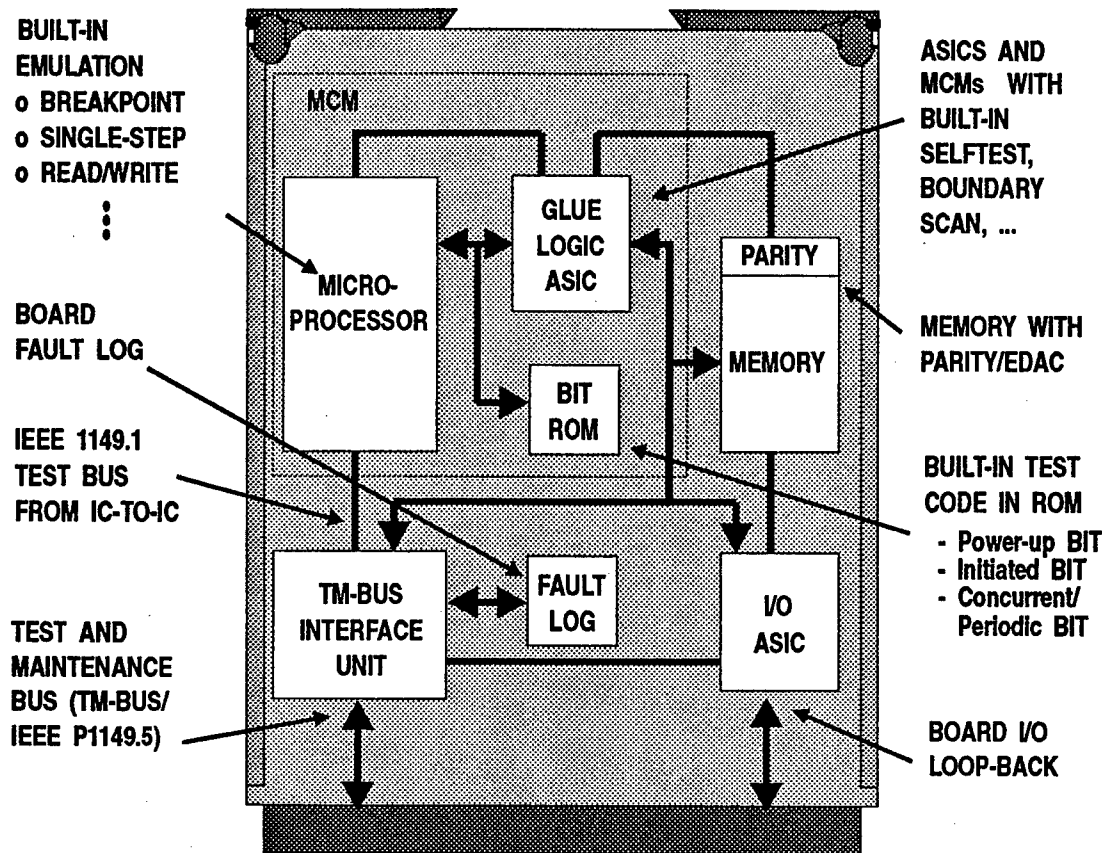


Figure 1.1.2-2. Module Test Bus interfaces to Ad-hoc BIT/BIST

1.1.3. Device Level Test Buses.

Another outgrowth of the VHSIC program was the development of the ETM-Bus. ETM-Bus adoption was limited to a few DoD contractors and lacked a common universal test command set. Other vendors had proprietary solutions and therefore prevented interoperability. IBM had its Level Sensitive Scan while TI used a proprietary bus called the System Maintenance Bus (SM-Bus).

During 1985, however, several European companies formed a working group to attempt to standardize an IC-to-IC serial scan test bus. The main drivers were increasing test cost, incompatible test features and the reduced physical access of high density designs. This group became known as the Joint Test Action Group (JTAG). The architecture of the JTAG bus was influenced by a test bus proposed by TI. This bus was then submitted to the IEEE for sanctioning and standardization and is known as the IEEE 1149.1, Standard Test Access Port and Boundary Scan Architecture. Before passing standardization, this bus went through several changes, including inputs from military and systems vendors to incorporate capabilities of the VHSIC ETM-Bus. This standard was passed in February, 1990.

The IEEE 1149.1 test bus provides a standard for boundary scan for device manufacturers as a way to offset testing difficulties predicted for board designs that used

circuitry, miniaturized packaging and advanced interconnect technology. Other device level testability features may be interfaced to IEEE 1149.1 as depicted in Figure 1.1.3.

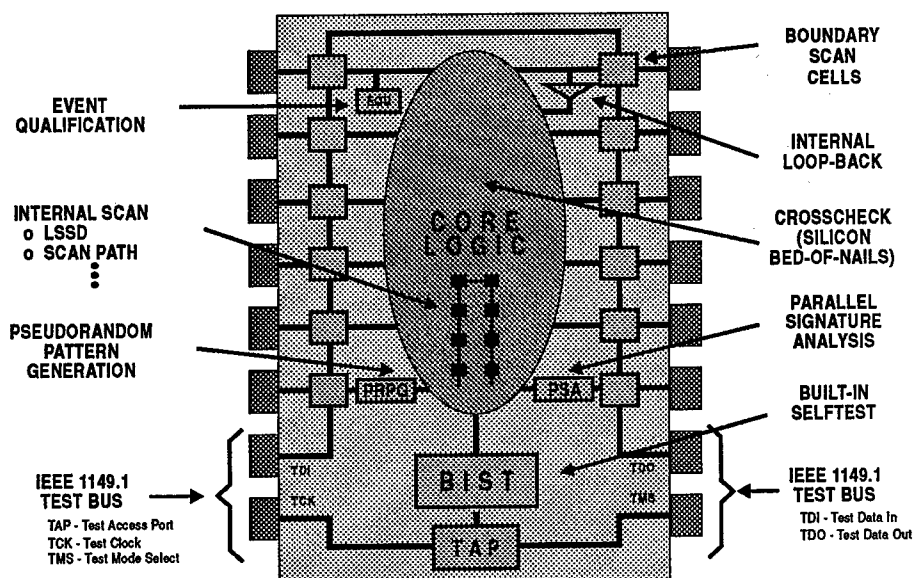


Figure 1.1.3. IEEE 1149.1 Interfaces to Device BIST

New device level test bus standards are emerging from the framework established by the IEEE 1149 committee. The Extended Digital Serial (P1149.2) and Mixed-Signal (P1149.4) test buses hold much promise. Extensions to the IEEE 1149.1, such as design and fault emulation, are just beginning to emerge as viable solutions. And, the standards for support languages (e.g. BSDL, SVF, etc.) and the test bus automation tools are coming available. The integration of test into the design process via H_t will have a major impact on the supportability of tomorrow's weapon systems.

1.2. IEEE 1149.1 / Joint Test Action Group (JTAG).

1.2.1. Overview and Intended use.

This standard defines a test access port (TAP) and boundary-scan architecture that can be implemented in integrated circuits (IC) to provide a method of testing interconnections in digital printed circuit boards or multi-chip modules.

The need for this standard was driven by technologies such as high-density integrated circuits, surface-mount packaging, and conformal coating of printed circuit boards. These technologies hinder traditional in-circuit test approaches.

1.2.2. Current Status.

This standard was approved as IEEE standard 1149.1 on February 15, 1990 and is designed into many commercial and defense projects.

Support by numerous hardware and software products shows its wide acceptance: Many major semiconductor vendors are marketing IEEE 1149.1 compliant chips, including controllers, memories, digital signal processors (DSP), buffers, latches, transceivers, bus monitors, microprocessors, specialized test logic, ASICs, FPGAs, etc. Both semiconductor (SC) and computer automated engineering (CAE) vendors have designed software tools supporting the standard.

1.2.3. Interface / Number of Pins.

The minimum number of wires/pins required by the standard is four, but an optional test reset wire/pin may be used.

TCK	Test Clock synchronizes bus operation
TDI	serial Test Data In pin accepts serial data
TDO	serial Test Data Out pin supplies serial data
TMS	Test Mode Select controls the state of the test bus
TRST	optional Test ReSeT, when low, asynchronously resets the TAP to Test Logic Reset and places the boundary scan register in normal operation mode

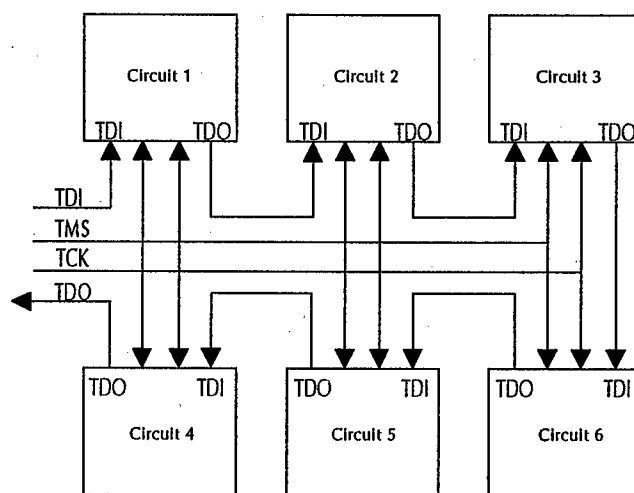


Figure 1.2.3-1. IEEE 1149.1 Ring Connection

Figure 1.2.3-1 shows a ring connection, which is the typical and intended style for this bus. A ring is formed by daisy-chaining TDI and TDO between ICs, while the control signals, TMS, TCK and, TRST if used, are bused out to all the slaves on a ring from the master or controlling source. A ring has two shortcomings: A fault on any ring connection will disable the entire bus; and, the scan path can become exceedingly long.

A star configuration, as shown in Figure 1.2.3-2, is an alternative style for this bus. It requires a separate TMS line for each target slave. Each target in the star is "enabled" by activating its associated TMS line. All TDO signals returning from the slaves may be joined, because TDO is tristated whenever shifting is not being performed. This style is tolerant to faults that effect the ring style.

In either configuration, only one TRST line is necessary to be fanned out to all slaves. TRST is highly recommended, because synchronous reset, through TMS and TCK, can take up to five clocks. Five clocks can be a long time if bus pins, controlled by the boundary scan register, are driving opposite logic levels. Please note that a power-up reset circuit, if available, may be included in an IC to reset the test logic asynchronously without a TRST pin.

The standard requires TMS, TDI and TRST to be pulled up to logic one inside the IC, because logic one is a safe value for each of these signals. A maximum of five consecutive logic ones clocked on TMS forces the TAP to Test Logic Reset. All ones clocked into the instruction register through TDI select the BYPASS instruction and normal operation of the IC. A high value forces TRST to be inactive. It is expected that TCK will be handled as a clock during board layout, i.e. properly routed and terminated, to maintain its integrity and to present a stable level if the TCK source is removed.

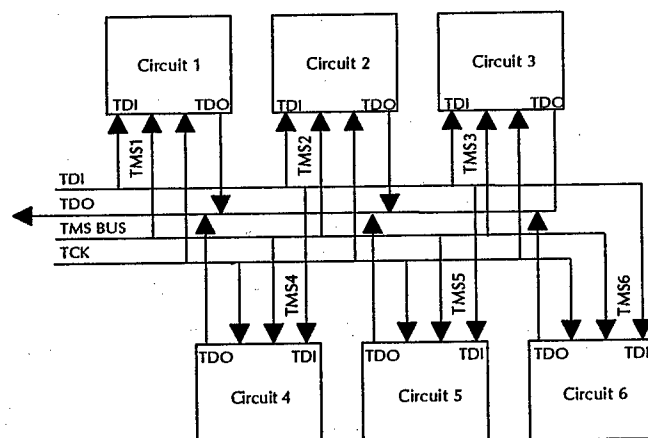


Figure 1.2.3-2. IEEE 1149.1 Star Connection

1.2.4. Architecture.

Figure 1.2.4-1 shows the minimum required architecture. The primary blocks are:

- TAP controller,
- instruction register,

- TDO selection,
- bypass register and
- boundary scan register.

Each block is described in the following paragraphs.

The **TAP controller** directs instruction scans, data register scans and test execution as described in the "Protocol" section. The italicized portions of the following text indicate states in the protocol diagram.

The **instruction register** controls data register selection and test operations, defined in "Instructions supported". It is loaded by shifting data in from TDI under control of the TAP. The instruction register must contain at least two bits, but has no limit on the maximum number of bits, although eight bits usually will suffice. During *capture-IR*, a logic one is preloaded into the least significant bit (LSB), and zero is loaded into the bit next to the LSB. (Note that the LSB is closest to TDO and that the most significant bit (MSB) is nearest to TDI.) Preloading both a one and a zero into each instruction register enables a flush test of the instruction path of the ring to check for both stuck-at-zero and stuck-at-one in the scan path. During *update-IR*, a latch either on the instruction register outputs or on its decoded outputs is loaded.

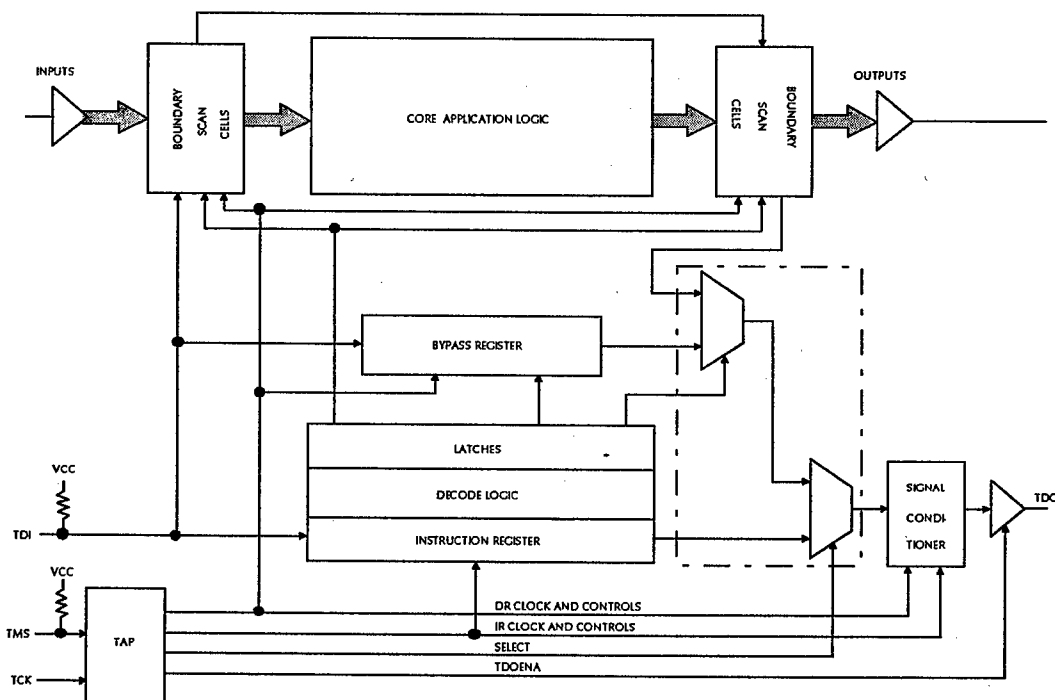


Figure 1.2.4-1. IEEE 1149.1 Architecture

TDO selection has two stages as shown in Figure 1.2.4-2. The outer stage, next to TDO, selects between the serial outputs of the instruction register and the data register path. It is controlled solely by the TAP. The inner stage selects the scan output of the selected data register, according to the value in the instruction register.

After reset, the TAP will be in the *Test Logic Reset* state and the instruction register will contain the BYPASS instruction. The BYPASS instruction causes the path between TDI and TDO during a *shift-DR* to select the **bypass register**. During the *capture-DR* state, the bypass bit preloads with logic zero. If the controller sends only ones out on TDO, the number of zeroes received before the first one is returned should indicate the number of slaves in the ring. The bypass bit needs no latch and is unaffected by the *update-DR* state. The main purpose of the bypass bit is to bypass one or more ICs during *shift-DR* to improve serial access to other ICs on the ring.

The **boundary scan register** is a shift register structure which forms a test collar around the ICs input and output signal pins. The register is composed of boundary-scan cells (BSC), one cell for each device functional pin and additional cells for each control output. BSCs are placed between the I/O buffers and the internal core logic of the device. In this position, they can either sample the state of the I/O for observation or drive the I/O to a known state for control. The BSCs are interconnected to form a serial bus (i.e. scan path or shift register) between the host devices' TDI pin and TDO pin. Data is shifted in and out of the cells under control of the TAP and the instruction. For a boundary scan instruction, such as EXTEST, SAMPLE or INTEST, the **boundary scan register** will be active and connected between TDI and TDO during *shift-DR*. Many variations of the boundary register are possible within the standard. As a minimum:

- all functional inputs, from uni- or bi-directional pins should be observable, and
- all functional signals which directly control the level or strength of outputs, whether two-state, three-state or bidirectional pins, should be controllable.

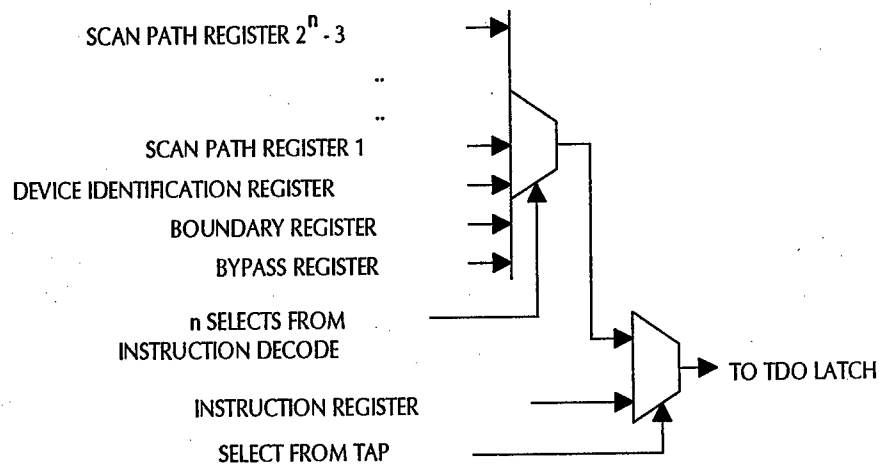


Figure 1.2.4-2. IEEE 1149.1 TDO Selection

A generic cell, depicted in Figure 1.2.4-3, which both observes and controls can be used for inputs or outputs. The elements of the generic cell are:

- an **input multiplexer** (mux) to select between observation or shift data,
- a **flip-flop** (FF) to be linked in the scan chain, which clocks data from the input mux,
- a **latch** on the FF output, which updates test control data at the end of each scan and
- an **output mux** to select between test control data from the latch or functional data.

The standard requires that test data presented to the system pins change only during *update-DR*. If test control values are shifted into the boundary register, a latch stage is needed to hold the output pin data stable until *update-DR*. Of course, during normal operation of the IC the system controls all functional (non-test bus) signals. Optional or user-defined tests generally require additional circuitry within the boundary register.

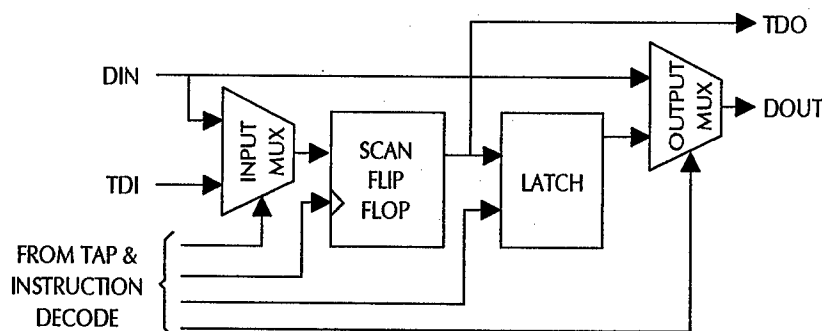


Figure 1.2.4-3. IEEE 1149.1 Generic Boundary Register Cell

Optional logic. Many forms of circuitry may be added to the minimum architecture, but they typically take the form of extra data registers. One optional data register defined by the standard is the device identification (ID) register. When the device ID register is implemented, the controller can confirm or interrogate the type of part(s) on the ring. ID data includes the manufacturer, part number and version or programming of a component. It must be 32 bits long, and its LSB (first bit shifted out) must preload with logic one. It is accessed through the IDCODE instruction. For devices without an ID register, the IDCODE instruction is decoded as a BYPASS. Therefore, if the first bit received from a data scan, when all devices are loaded with the IDCODE instruction, is a logic one, the next 31 bits will identify the first component. Then, the 33rd bit returned will start the next IC. Alternately, if the first bit is a zero, that device is in bypass, and data from the next chip will start with the second data bit returned. So, the controller can blindly determine what types of parts are on the ring, even when only some have identification. However, adding the ID register is highly desirable. If gate count is questioned, consider this: It is perfectly legal to share logic for the device ID register with

another register, such as the boundary register, as long as the contents of the output latch of shared cells is not changed.

All other test data registers are design-specific. Their access must conform to the standard, and their length must not change for different instructions. Otherwise, their design is unrestricted. Design-specific test data registers generally enable either control of specialized test functions or access to internal functional data. Control registers may contain individual control bits, expanded commands, values to program a linear feedback shift register, counter values, counter control, test protocol, seed data, etc. The wide range of special functions demonstrates the flexibility of test access available through the test bus. Operation of the tests listed for control register data can be quite elaborate, and involve surrounding components for board level testing. Testing core logic can be enhanced simply by adding scannability to any number of internal functional registers. The extension section on internal scan covers this topic.

1.2.5. Protocol.

The sixteen-state TAP, as stated in the interface section, is controlled synchronously by TMS and TCK, and may be reset asynchronously by the optional TRST signal. The TAP state diagram, defined by the standard, is shown in Figure 1.2.5. The loopbacks in the diagram identifies six stable states. These stable states allow

- resetting the test logic (Test Logic Reset / STRAP),
- remaining in idle or running tests (Run Test / Idle),
- scanning a data path (Shift-DR),
- scanning the instruction path (Shift-IR),
- pausing during a data scan (Pause-DR), or
- pausing during an instruction scan (Pause-IR).

The *test logic reset state* is described in the Interface / Number of Pins and the Architecture sections.

The scan states allow serially accessing the selected data register scan path. For *shift-IR*, the scan path always is the instruction register. For *shift-DR*, the scan path will depend on which data register is selected by the current instruction, as stated in the TDO selection block of the Architecture section. Any non-selected scan path will hold its value, excluding the bypass register.

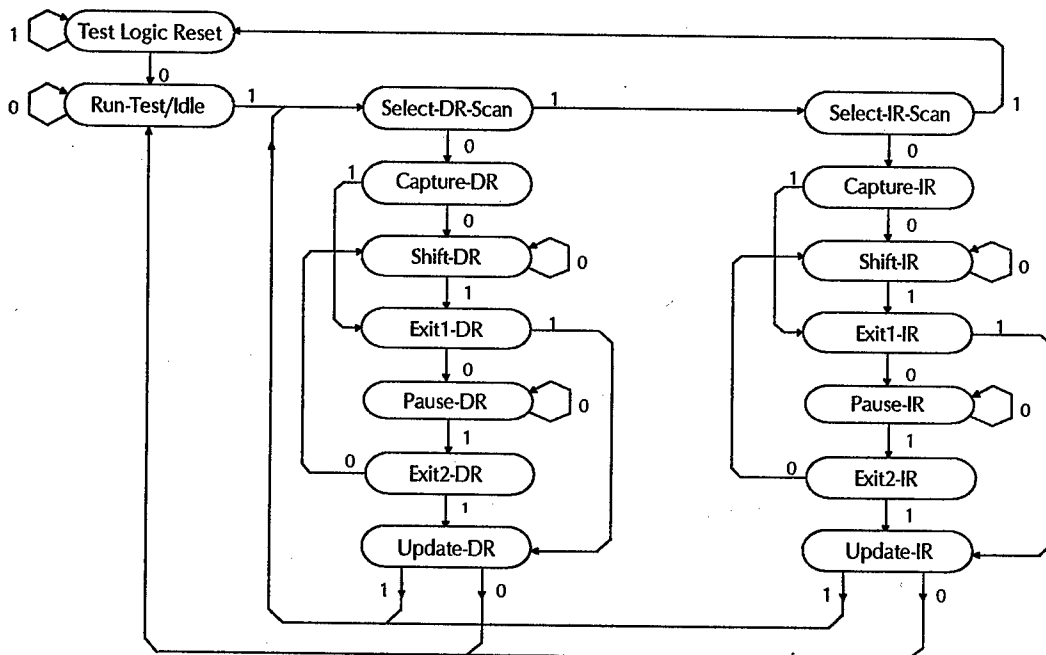


Figure 1.2.5. IEEE 1149.1 Test Access Port Protocol

During the *run-test / idle* state, the test loaded into the instruction register may execute.

The *pause-IR* and *pause-DR* states enable pausing data transfer between the test bus data and slave. For example, a microprocessor used to run a scan test may force the TAP to pause anytime during either (data or instruction) scan, if it needs extra time to examine or store incoming data or fetch the next outgoing data.

Four of the temporary states, capture and update for both data and instruction scans, are also critical. During capture, status or test result data may be preloaded into the instruction register or the selected data register, respectively, prior to scan. During update, data in the shift register is parallel loaded into output latches. An end-of-scan latch-update prevents control data from rippling through the circuit during scan.

At least one prior bus, VHSIC ETM, also specifies boundary scan, but the protocol is quite different. Previous buses provide fewer states and require multiple control signals, at least two. In some cases, a data line doubles as a control signal. The protocol of the state machine inside the IC, prevents illegal bus operations. For example, it is impossible to change directly from *shift-IR* to *shift-DR*, because the protocol of the state machine does not allow such a transition.

1.2.6. Instructions supported.

The boundary scan logic is responsive to control input from the TAP controller and the instruction register decode to execute a variety of standard and user definable test mode instructions. The standard defines 7 public instructions, of which only 3 are required.

BYPASS	Required
EXTEST	Required
SAMPLE/ PRELOAD	Required
INTEST	Optional
RUNBIST	Optional
IDCODE	Optional
USERCODE	Optional

BYPASS.

BYPASS is a required instruction. During BYPASS, the boundary test logic is disabled; input and output signals flow freely through the test collar, enabling normal operation of the IC. Functional data is neither controlled by the boundary cell output muxes nor observed through their input muxes. BYPASS is comparable to a no-operation "NOP" instruction, because no test is performed.

As defined in the architecture section, during BYPASS, the bypass register is selected during *shift-DR* operations. Putting a device in bypass mode effectively removes it from the data scan path, by limiting its scan path to one "inert" bit.

EXTEST.

EXTEST is a required instruction. During EXTEST, the boundary test logic is enabled and placed in its external test mode; the test collar provides an input response and output stimulus structure to enable testing of interconnects between multiple board-resident ICs, via serial access operations. The response is captured through the input muxes of boundary register cells on IC inputs; and the stimulus is provided through the latch and output mux combinations of boundary register cells on IC output level and control signals.

EXTEST is a boundary scan instruction, and, as such, activates and places the boundary scan register in the path between TDI and TDO during *shift-DR*. Response data is loaded during *capture-DR* and stimulus data is output during *update-DR*. The response data scanned out is from the stimulus data loaded during the previous scan operation. Thus, the first scan returns old data as it loads the first data to apply, and the final scan loads "dummy" data as it reads the last data to observe.

SAMPLE/PRELOAD.

SAMPLE/PRELOAD is a required instruction. When the boundary test logic is enabled and placed in its non-intrusive sample mode, the test collar allows the IC to function while data entering and leaving the IC is sampled and shifted out for examination via the boundary scan register. For the sample function, it does not necessarily matter what data is scanned into the boundary register, because the sample is meant to observe. However, the preload function is intended to place the boundary register in a predetermined state for

later tests. The two functions can be executed simultaneously, if that suits the purpose of a given test.

SAMPLE/PRELOAD is a boundary scan instruction, and, as such, activates and places the boundary scan register in the path between TDI and TDO during *shift-DR*. Sampled data is parallel loaded during *capture-DR* and is shifted out during *shift-DR*, while preload data is shifted in.

INTEST.

INTEST is an optional, but recommended, instruction. When the boundary test logic is enabled and placed in its internal test mode, the test collar provides an input stimulus and output response structure to enable testing of the ICs interior logic, via serial access operations. The response is captured through the input muxes of boundary register cells on IC output level and control signals; and the stimulus is provided through the latch and output mux combinations of boundary register cells on IC inputs. The standard requires control of all non-clock input pins. Control of clock signals may be delegated to the board or system design. INTEST may operate identically to EXTEST, such that the only difference is how the controller views the test data, unless the chip requires special test logic to enable single step operation. During INTEST it is allowable to globally tristate all 3-state and bidirectional pins on the chip, to guarantee it remains isolated from surrounding devices, regardless of the values loaded into scan latches during update.

INTEST is a boundary scan instruction, and, as such, activates and places the boundary scan register in the path between TDI and TDO during *shift-DR*. Response data is loaded during *capture-DR* and stimulus data is output during *update-DR*.

RUNBIST.

RUNBIST is an optional instruction. Using this test mode, the ICs interior logic can be quickly verified using user-defined BIST circuitry and algorithm, reducing, or even eliminating, the need for time consuming single-step testing through INTEST. Because RUNBIST does not single-step, it may perform at-speed testing to check not only the logical function, but also timing through delay paths inside the IC. Like INTEST, the outputs must be controlled to safe values throughout the test. The BIST controller is responsible for initializing data prior to running the test. If the boundary register is used for RUNBIST, it may be setup with a SAMPLE/PRELOAD instruction. The BIST controller also is required to stop the test automatically, without external input. Selftest results shall be loaded into a selected data register (possibly the boundary register) during *capture-DR*.

RUNBIST implementations are possibly as diverse as ICs incorporating this instruction, but their operation from the TAP is very regular, as required by the standard. See the BIST extension section.

IDCODE and USERCODE.

The IDCODE instruction is described as part of the optional logic under the architecture section. A companion instruction to IDCODE is USERCODE. When a user-programmable device contains a device identification register, and its programming cannot be determined through other test logic, the USERCODE instruction may be used. When executed, USERCODE loads a binary code, selected by the component designer, into the device identification register, and activates this register to be scanned out for observation of the programmed function.

Through the addition of new instructions and internal scan registers or test logic almost limitless test functions can be performed. Since all the test capabilities provided by the boundary scan test standard are enabled via the TAP interface, they can be reapplied after the IC has been embedded in the system. Thus, tests created at each level of development can be reused through the entire product hierarchy and life cycle.

1.2.7. Application.

Several features of IEEE 1149.1 make it easy to apply powerful test techniques:

- Virtual test points enable partitioning circuits at any level.
- Test bus interface and structures are compatible with many test methods.
- Test bus interface enables reuse of test throughout hierarchy and life cycle.

When applying the test bus to a project, a few considerations should be made during the process of designing test logic and test programs.

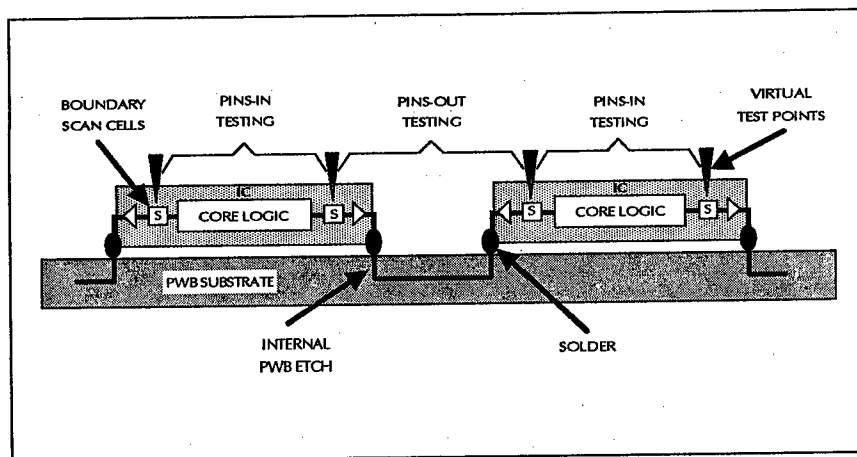


Figure 1.2.7-1. Pins-In and Pins-Out Testing using IEEE 1149.1

Virtual test points and partitioning.

Previous industry testing of populated boards was performed by test fixtures utilizing a "Bed-Of-Nails" test approach. These test fixtures were costly and reacted to change in board designs very slowly. IEEE 1149.1 replaces these expensive test fixtures by utilizing a concept of "Virtual Test Points" for Pins-In and Pins-Out testing.

Pins-In testing (associated with INTEST) is accomplished by loading test patterns at device inputs, driving data across device core logic, and capturing patterns at device outputs. Pins-In testing verifies IC core logic. Pins-Out testing (associated with EXTEST) is accomplished by loading/driving test patterns at device outputs and capturing patterns at the next device's inputs. Pins-Out testing, thus, checks the pin-to-pin wiring interconnects between devices on a board. Figure 1.2.7-1 illustrates the concept of pins-in/pins out testing. Note that virtual test points are not subject to the potential noise problems that may be caused by additional etch and pins of test points.

Virtual probing does not eliminate the need for some sort of At-Speed testing to verify the interactions between devices executing at their normal speed. It is preferred that at-speed testing be performed by Built-In Self-Test (BIST) applications. BIST is also advisable in place of Pins-In testing of IC core logic, due to the amount of test patterns and test time generally required for INTEST.

By using generic BSCs, isolation of the device can be achieved. Besides isolating the chip from the board or system, use of IEEE 1149.1 compatible components enables partitioning a system into more testable subunits. An illustration of module functional partitioning to increase testability is shown in Figure 1.2.7-2.

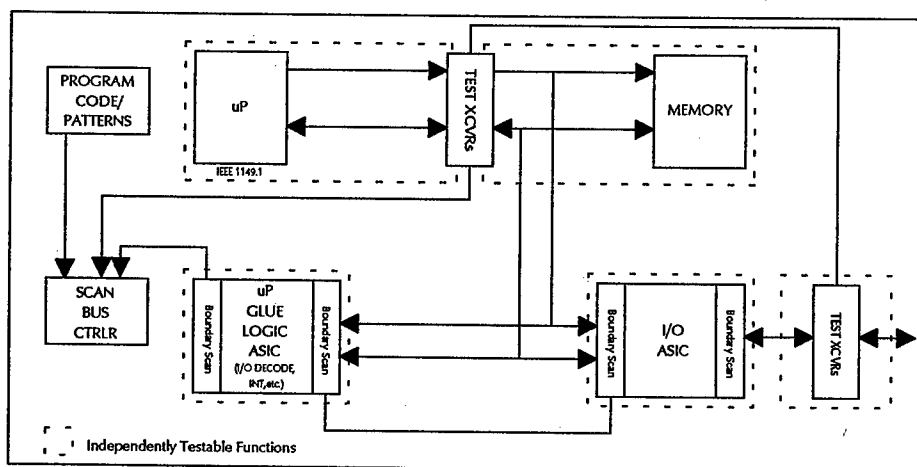


Figure 1.2.7-2. Module Partitioning for Testability

Virtual test points can be created by scan cells other than BSCs. A BSC is a special, albeit required, type of scan cell. The scan cell is the basic building block of the IEEE 1149.1 architecture. When scan cells are used internal to the ASIC, internal nodes also become virtual test points. By partitioning the ASIC into functional blocks, and by properly placing scan cells around these partitions, sections of the circuit may be tested separately or in parallel. Figure 1.2.7-3 illustrates this concept.

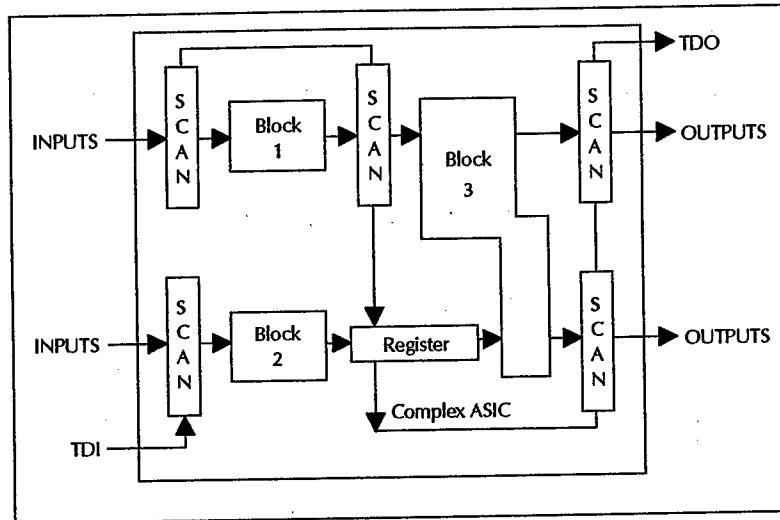


Figure 1.2.7-3. IC Partitioning For Testability

Test Methods Achieved through IEEE 1149.1.

The IEEE 1149.1 boundary-scan standard provides designers with capabilities that will make their digital designs more testable and producible. Virtual test points can be used along backplanes, around processors or memory, around functional blocks and surrounding analog logic. Use of scan in each case has special considerations and benefits.

Backplane Testing

Until recently, typical board to board testing relied on Built-In-Test to detect backplane faults; open etch, faulty connector pin conditions, and short-to ground. The results were not always good because backplane signals were controlled and observed through functional logic and it was difficult to limit the size of ambiguity groups. However, with the emergence of IEEE 1149.1, backplane testing for these board to board faults can be achieved with favorable results.

By surrounding the bus interface of a board with boundary-scan cells, as illustrated in Figure 1.2.7-4, backplane faults can be effectively detected and isolated. Functional logic is not used to test the external signals which eliminates that logic from the ambiguity group. Each board in turn can drive logic signals onto the backplane while the other boards within the system receive the driven logic values.

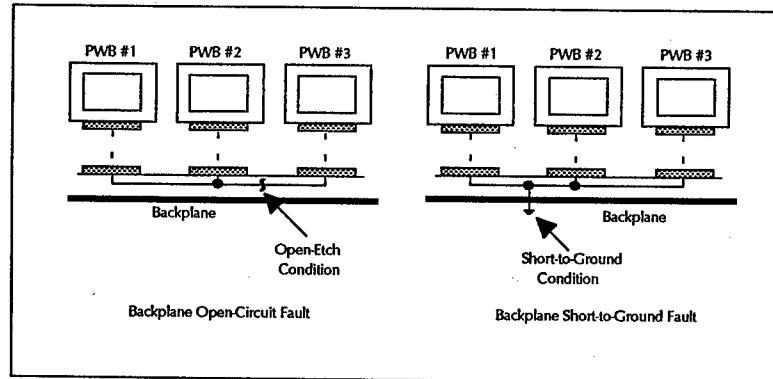


Figure 1.2.7-4. Backplane Testing using IEEE 1149.1

Processor and Memory Testing using IEEE 1149.1

Microprocessors with embedded IEEE 1149.1 allow for testing interfaces to which they are connected. Memory access and I/O interconnect testing is enhanced through proper positioning of scannable cells within the board design. Typically, embedded tests begin using a kernel (or start-small) approach which requires a significant amount of hardware (processor, memory, etc.) to be functional in order to execute. Boundary scan can be used to reduce the kernel to a smaller ambiguity group and therefore result in better isolation.

Large memories can be partitioned into more testable arrays, thus allowing for faster test times and better isolation of each memory partition. This is illustrated in Figure 1.2.7-5. Boundary-scan cells positioned within ASICs provide for controllability and observability of memories which otherwise would not be accessible. Scan cells on memory address, data, and control interface logic allows for reading/writing of memory locations without processor involvement. It is not recommended that all memory cells be verified via boundary scan, only the address, data and control signals needed to access the memory. Memory testing via IEEE 1149.1 devices could be further enhanced through incorporation of pattern generation and data compaction devices within the board design. Otherwise, conventional processor-based memory test algorithms should be used to verify memory cells or contents.

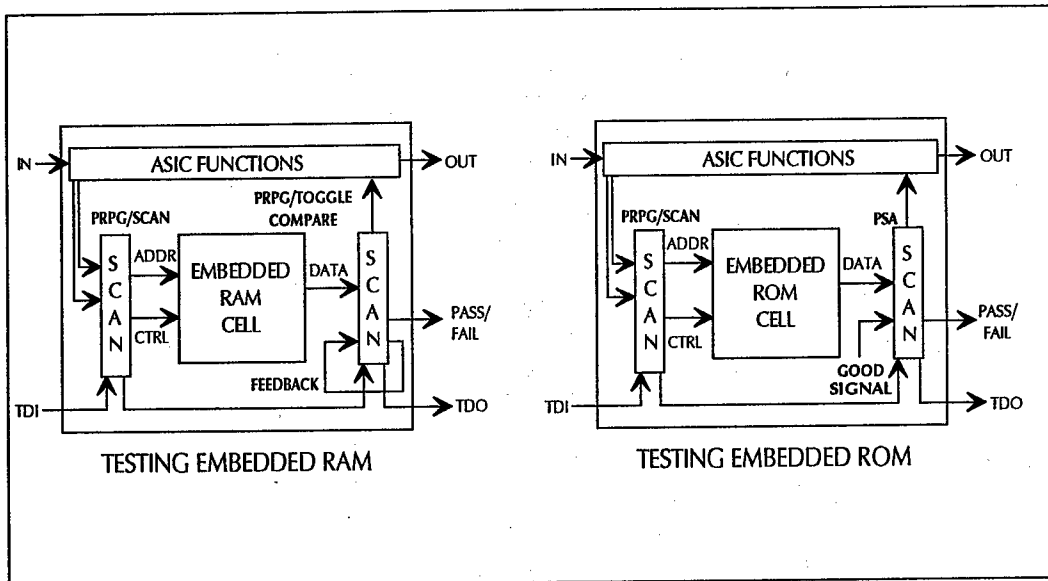


Figure 1.2.7-5. Testing Embedded Memory

Board Functional Logic Testing.

Within many complex designs, composed of VLSIs and ASICs, exists discrete logic which mates functional partitions on the board. Very often this discrete logic does not include IEEE 1149.1 technology. Testing this logic can be as simple as surrounding this discrete logic with scannable devices, scanning test vectors into the boundary-scan device, driving the vectors across the logic, and capturing the resultant data at the boundary of the next scannable device. Another method for testing discrete logic is the use of scannable devices which perform pattern generation and data compaction applications. Using these devices in the above scenario results in faster test execution.

Analog Testing.

The primary thrust of the IEEE 1149.1 boundary-scan architecture was to reduce the complexity of testing digital circuits. However, the use of boundary-scan also has benefits in a mixed analog/digital environment. By providing scan-based logic between digital signals received from or supplied to analog logic, improved testability/ isolation between analog and digital circuitry can be accomplished. This is illustrated in Figure 1.2.7-6. Care must be taken to ensure that signals driven to and from the A/D and D/A logic do not ripple during shifting of the boundary scan register.

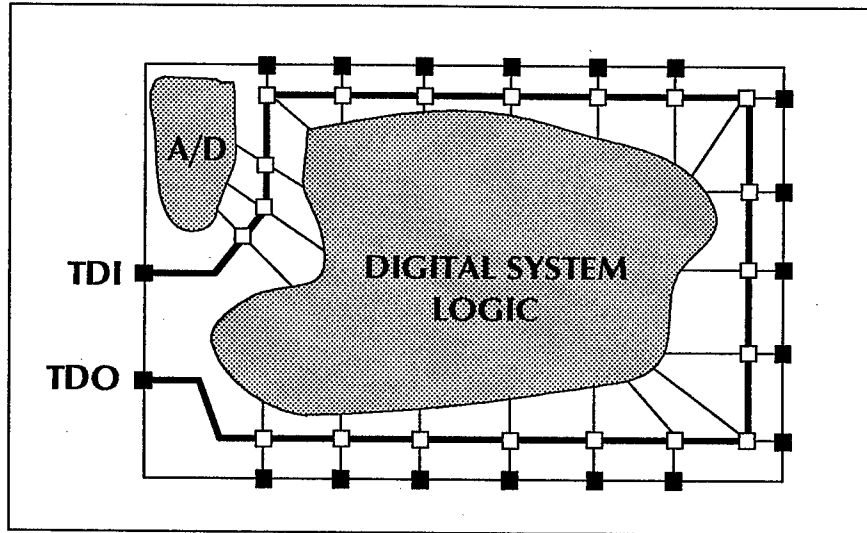


Figure 1.2.7-6. Testing Analog logic

Hierarchical Testing.

ASIC Verification and Production Testing.

Current design and manufacturing processes are approaching a point where "in-circuit" probing of components is no longer feasible. IEEE 1149.1 enhances the designer's ability to verify a design by allowing access to internal nodes and embedded test features. Access to these test features can be accomplished via low-cost debug/test stations which can be PC-based (see Figure 1.2.7-7). Software tool-sets employed on these stations provide test access, thus allowing faster debugging and validation of designs. The IEEE 1149.1 test bus architecture improves the device production cycle by allowing simpler, inexpensive and portable test fixtures and re-usable tests. Test vectors used during design verification can be reused during production testing; thus reducing production test development costs.

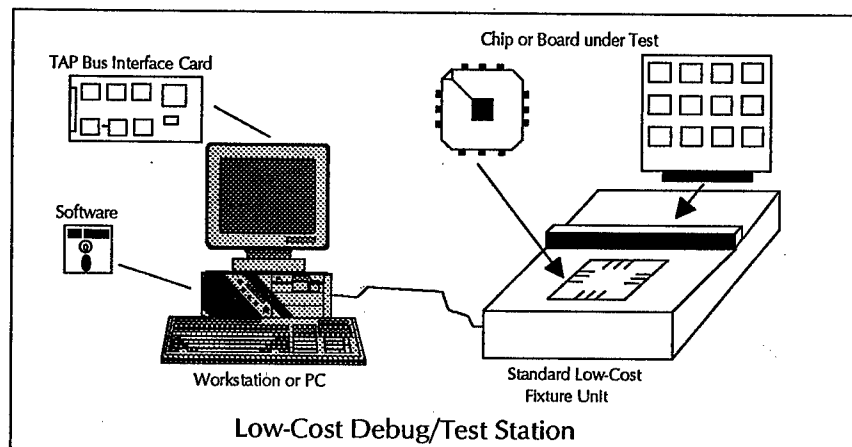


Figure 1.2.7-7. PC Based Test Stations

Board Verification and Production Testing.

IEEE 1149.1 provides a method to observe and control circuits without the need for clips or probes, thereby making board verification easier. Via the IEEE 1149.1 test bus interface the designer is able to control the state of the design, setup specific conditions, and observe how the design responds to test stimulus; all under software control. Portions of the design can be individually tested in the absence of other portions, thus allowing early verification of the design prior to making final design decisions.

With the emergence of IEEE 1149.1 technology, production testing of populated boards can take a new direction, away from expensive test fixtures, and toward less expensive personal computers and desktop fixtures (illustrated in Figure 1.2.7-8). Production testing can take advantage of the improved visibility and control, at-speed testing, and fault detection/isolation capabilities of IEEE 1149.1 to implement a low-cost manufacturing defect testing program. This low-cost testing program can be used for subsequent board level tests in the factory, field, and depot.

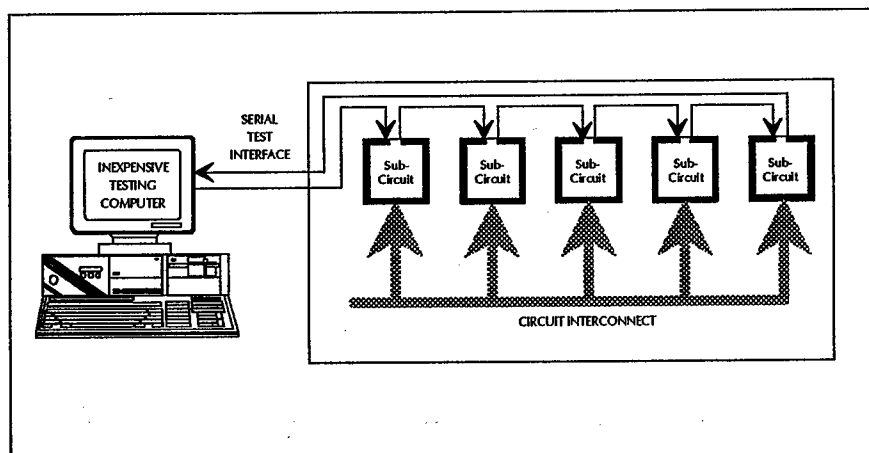


Figure 1.2.7-8. Board-Level Testing System using IEEE 1149.1

Process.

Design Concerns.

When a design is considering the use of IEEE 1149.1, whether for system, module, or device, consideration must be given to the architecture of the scan path. Scan bus design must be a structured, parallel effort, and one which fosters close cooperation between the design and testability engineers.

Consideration must be given to what functional blocks of logic inside an ASIC will be tested by internal scan, what functional blocks of logic on a board will be surrounded by IEEE 1149.1 devices, number of parallel scan paths required, which circuits will be on each scan path, and what signals need to be controlled by IEEE 1149.1.

Board Design.

In most cases, the normal buffers and transceivers within the design can be replaced with IEEE 1149.1 scannable cells, thus providing controllability and observability of each functional block. I/O interfaces, memory, address, data, and control signals can also be surrounded by IEEE 1149.1 scannable cells, and thus allow testing of these elements via the board embedded IEEE 1149.1 test bus.

Medium-to-high complexity modules should have adequate partitioning to allow independent testing of major logic functions. In addition to performing buffer, latch, or transceiver functions, scannable cells can be controlled via the IEEE 1149.1 test bus to load or sample signal states, thus enhancing partition isolation, and reducing ambiguity among functions. By proper module design partitioning, provided by IEEE 1149.1 scannable cells, module failures can be detected and isolated with less probing or part substitution, and therefore can result in repair and replacement savings.

Clock skew.

While the standard is designed to prevent hold time errors at the board or system level, it is still recommended to route TMS and TCK together during board layout. Potential timing problems inside the IC are not covered by the standard and must be solved by the engineer. Several approaches are commonly used for shift register design.

- For the boundary register, which often is the longest, the clock can be routed from the BSC nearest TDO to the BSC closest to TDI. However, this may require manual intervention during IC layout.
- Some ASIC libraries offer special scan cells with built-in delay on the shift input. These cells work well and their use may be completely automated.
- In a master/slave flip-flop, used as a scan cell, the master side would latch the serial input on the rising clock edge and the slave side would latch the serial output on the falling edge. The master/slave flip-flop effectively uses the same technique as defined by the standard for the board level.
- If users build shift registers from ASIC library hard macros, they may consider multi-bit shift register macros, which both minimize clock skew and, through buffering, lower clock loading.

While use of a serial test approach magnifies the hold time problem, it is not new as reflected by the many solutions available.

ASIC Simulation Concerns.

ASIC design simulation is more easily achievable within designs which are captured hierarchically. Test patterns used during the simulation process should be developed for application at the functional level. These test patterns then become transferable to higher levels within the hierarchical test structure. When generating test patterns to be applied at the ASIC level, insure the patterns do not apply test stimulus to the normal I/O pins while at the same time applying test stimulus to the scan test cells. When they are applied together your responses may be unpredictable.

Vector Generation.

ASIC test stimuli consist of parallel and serial test vectors; parallel test vectors for testing the normal ASIC operations, and serial test vectors for testing the IEEE 1149.1 test logic. The parallel test vectors should be generated first in order to verify that the test logic is not interfering with the normal ASIC operations.

1.2.8. Bus timing.

Clock frequency. The simplicity of the protocol requires little setup time for decode. However, to prevent hold time errors, TDO and TMS-out are output on the falling edge while TDI and TMS-in are latched on the rising edge. Forcing the output delay plus the input setup times to fit within half a clock cycle effectively halves the maximum possible frequency. The standard does not specify a maximum frequency, but does stipulate that the interface must function when TCK is halted.

Throughput. Once a scan is started, it may continue for as many clocks as necessary. There are no packet constructs with time-consuming frames. As long as the controller can monitor and provide data during shift operations, it never needs to force the bus to pause.

1.3. IEEE P1149.2 Extended Digital Serial Subset.

1.3.1. Overview and Intended Use.

The goal of the P1149.2 standard is to develop a test architecture supporting test and diagnosis of digital devices at all levels with the following characteristics:

- Implementation is flexible, and requires few structures.
- User may add test features easily.
- Limits impact on performance or cost of design.

Generally, the P1149.2 working group wants this standard to support boundary scan, internal scan and user-defined tests, while imposing minimal test architecture requirements, i.e. limiting gate count.

1.3.2. Current Status.

The P1149.2 working group is still active, but the proposal is not ready for ballot. The latest draft is number 0.2 and is dated February 15, 1991. Currently, both the 1149.1 and P1149.2 working groups are examining the possibility of buses coexisting. The direction is towards one or more test pins separate from the buses, which would select the active bus.

1.3.3. Interface / Number of Pins.

The minimum number of wires required by the standard is five; but it is not fixed.

TCK	1 or more pins	Test Clock. TCK may comprise multiple phases (signals), but TCK must be separate from the system clock.
TDI	1 pin	serial Test Data Input
TDO	1 pin	serial Test Data Output
STM	2 or more pins	Select Test Mode inputs select the test function. STM0 and STM1 are required, but additional inputs are allowed.

Edge-Triggered or Multi-Phase TCK. Both the scan path and BIST logic may be clocked by either an edge-triggered or a multi-phase test clock (TCK). A possible reason for allowing multi-phase clocking may be compatibility with Level Sensitive Scan Design (LSSD) on internal scan paths. If a multi-phase clock device adheres to the proposed standard, it may be linked in a chain with an edge-triggered clock device. The controlling source will need to generate both types of clocks in proper synchronization. Whether multi-phase or edge-triggered clocking is selected, this draft of P1149.2 still requires TCK to be different from the system clock.

1.3.4. Architecture.

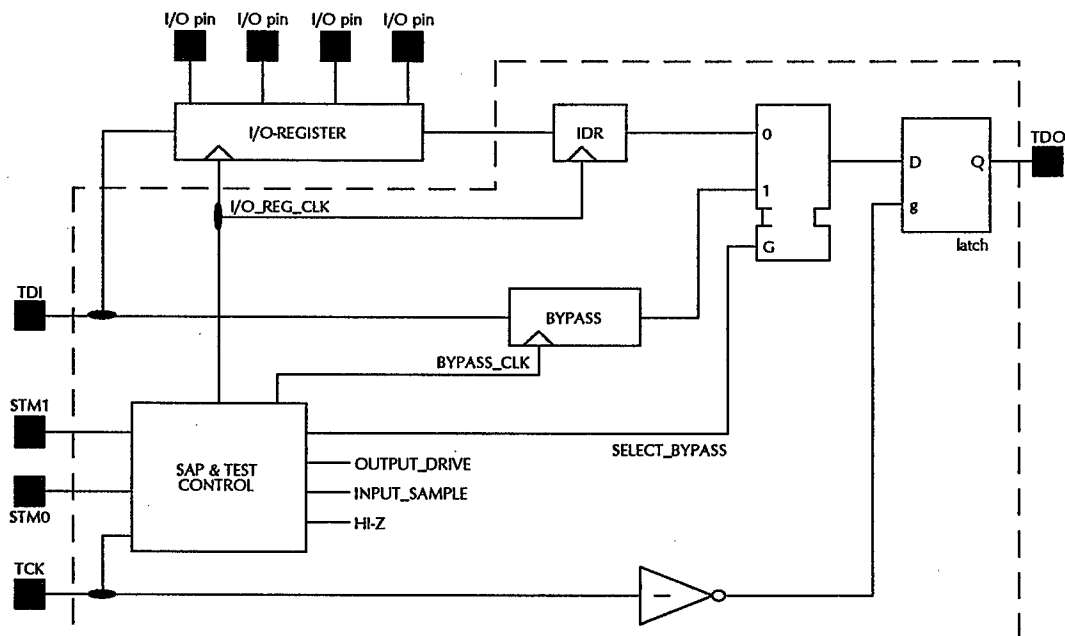


Figure 1.3.4-1. Minimum P1149.2 Architecture

Four major elements comprise the minimum P1149.2 architecture, as described in draft 0.2, dated February 15, 1991:

- I/O-Register
- Scan Access Port (SAP)
- Implementation Detail Register (IDR)
- Bypass Register

They are shown in Figure 1.3.4-1.

I/O-Register. In the draft version, boundary scan is achieved through an I/O-register. The I/O-register, like the IEEE 1149.1 boundary register, is meant to test interconnections between ICs on a board or multi-chip module. Unlike a boundary register, an I/O-register may share register cells with the core logic of the device. Because IEEE P1149.2 does not require the I/O-register cells to be dedicated test logic, these cells cannot perform test functions without disrupting normal operation. Note that the data path still gains a propagation delay, through the mux to select between normal or shift data. Also, because the parallel-update stage is not required, output pin values sourced by the I/O-Register may toggle during register shift operations. When toggling is a problem, P1149.2 frees the user to choose any workable solution, with or without latches.

SAP. Test control will come from a simple decode of two or more Select Test Mode (STM) control input pins. Letting n be the number of control pins, the SAP combinationally decodes n to 2^n . Eliminating the TAP state machine and sharing functional logic are planned to reduce gate count overhead. See Figure 1.3.4-2.

1.3.5. Functions Supported.

P1149.2 has neither protocol nor instructions, but it does have required functions, which are decoded from the STM inputs.

STM1/ STM0	Test mode	Required Functions	Optional Functions
0/0	I/O-DRIVE/ SAMPLE	output-drive, I/O-sample, bypass	input-drive
0/1	USER	none (and bypass)	update, serial internal-scan, special user functions
1/0	I/O-SCAN	I/O-scan, high-impedance	parallel internal-scan
1/1	SYSTEM	regular system (and bypass)	I/O-sample, parallel internal-scan

Support of additional functions requires additional STM inputs.

1.4. IEEE P1149.4 Mixed-Signal Test Bus Standards.

1.4.1. Overview and Intended use.

The intention of P1149.4 is to define a mixed-signal test bus for use at the device and assembly levels enabling control and observation of analog signals and supporting built-in test structures. Use of the standard should shorten test development time and lower test costs, but raise test quality. P1149.4 does not plan to solve all mixed-signal test problems.

1.4.2. Current Status.

The working group has developed a strawman approach and divided the research among its members. The strawman resulted from a combination of proposed frameworks and a determination that interconnect testing is a primary concern. The level of interest and participation in the current study indicates that a proposal may be ready by the end of 1993 or early 1994.

Currently, requirements are at a high level.

- Address test needs at device and board levels.
- Observe signals.
- Control signals.

- Minimize overhead.
- Possibly compensate for performance degradation.
- Provide parametric analysis with minimal disturbance of sensitive nodes.
- Be compatible with 1149.1 and other 1149 standards in progress.

Operating procedures. The working group holds quarterly meetings. They have evaluated technical frameworks presented by working group members for consideration as methods for mixed-signal test and design-for-test. International Microelectronics Products (IMP), AT&T and Gould/AMI have shown chip-level frameworks. From these presentations and inputs from all levels of the mixed-signal market, they have distilled the general requirements and scope of the test bus. A common framework for board and chip levels is under investigation by a diverse group, including industry and academia. Variations and elements of the proposed solution are being examined for specific features and tolerances needed for a useful specification. New members are sought to expand the research to insure a comprehensive and viable approach is obtained. When all avenues of implementation are sufficiently addressed, the issues and alternative approaches will be compared and condensed into a cohesive standard proposal.

1.4.3. Architectural Elements of Proposed Frameworks.

Presented solutions, which contributed to the strawman, have three main features:

- a 2-wire analog test bus
- digitally controlled switching networks to isolate analog circuitry for test
- some type of in circuit test (ICT)

Analog test bus.

The 2 wires in this bus have been called:

ATDI ATDO	Analog Test Data In Analog Test Data Out	TDI /TDO are compatible with IEEE 1149.1.
AB1 AB2	Analog Bus 1 Analog Bus 2	These could both be bidirectional signals.
ASI ASO	Analog Scan In Analog Scan Out	Analog Scan is a misleading terminology.

Whatever they are called, they are effectively the same, but for convenience ATDI/ATDO will be used here. Only one analog test input is used, because it is impractical to control multiple analog signals. The variety of waveforms needed to test all types of analog circuitry and a question of how many inputs is enough drives this decision. Also, it is reasonable to assume that a functional test can detect board or system failures, so that the

analog test bus is needed primarily for isolation. Isolation may be achieved with one stimulus and one measurement channel. ATDO provides a movable virtual probe to observe one internal signal at a time, until the fault is isolated. EXTEST functions can be performed on analog pins by connecting ATDI to each analog output pin in turn and observing the result through the ATDO test point for the input pin on other circuits. ATDI and ATDO are meant to be driven and observed by an external tester. As a result, the type of signals provided or measured are limited only by the capability of the ATE. External ATE should be able to measure parametrics, such as voltage, current, power, amplitude, phase, frequency, etc.

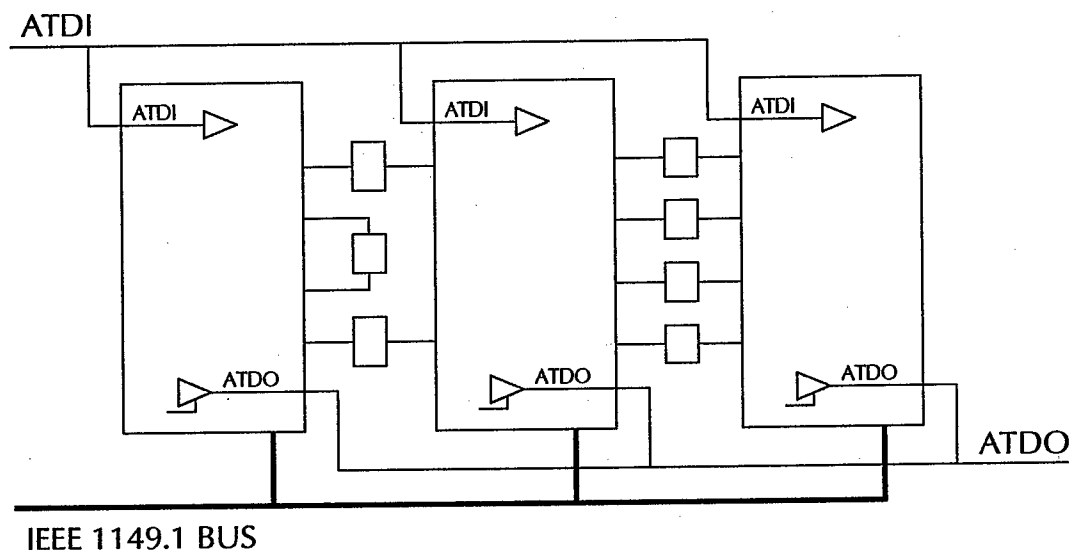
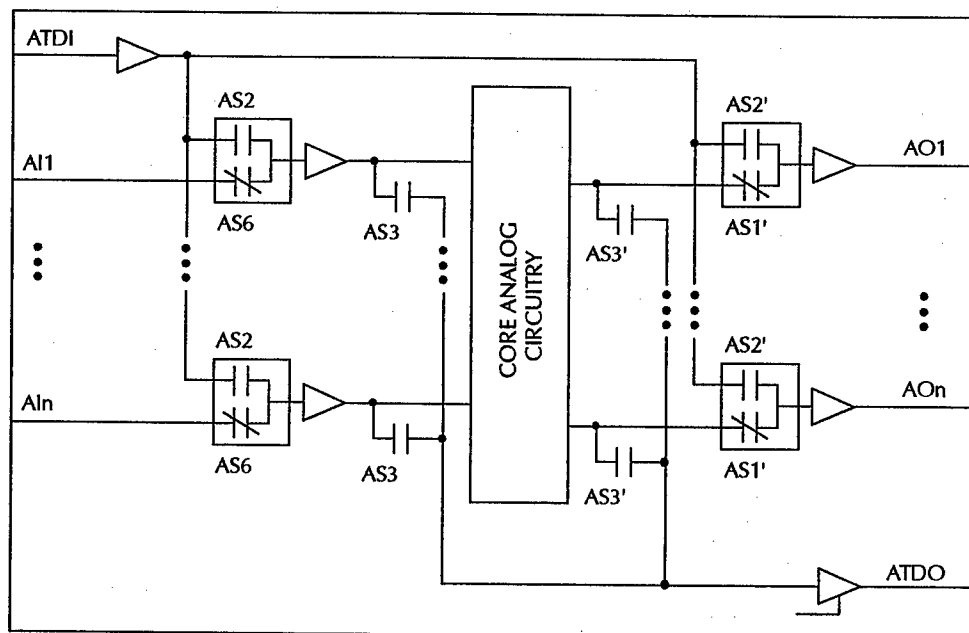


Figure 1.4.3-1. Parallel Connection of Analog Test Bus

ATDI and ATDO will be connected in parallel, as shown in Figure 1.4.3-1, and will not be daisy-chained. If they were daisy-chained, every device's ATDI/ATDO would have to pass the maximum and minimum parametrics existing on the chain. With a parallel connection, each device only needs to protect itself from signals outside its tolerances. To enable merging ATDO from every device, it must be a tristate signal. Tying together many switches without tristate buffers might slow the frequency response. The standard must deal with capacitance and resistance at the board level. A solution might be to set minimum input impedance and maximum output impedance standards at the chip and board level for the bus, requiring judicious use of buffer amplifiers at inputs and outputs, to ensure acceptable (and standardized) fan out and fan in capabilities. Input to each chip from ATDI is buffered to avoid bus loading and capacitance effects. ATDO from each chip is also buffered to minimize bus loading and capacitance effects and to minimize effects of switch resistance.

Switching networks.





LEGEND: Open switch 
 Closed switch 

Figure 1.4.3-2. Minimum Implementation of Switching Network

The analog test bus is useless without a switching network to select signals for control or observation. The switching network is composed of analog muxes, switches and/or transmission gates which are controlled by digital signals. A minimum implementation, shown in Figure 1.4.3-2, would require the functional equivalent of 3 switches and 2 digital control signals for each device pin. (The switch numbers, AS_n , correspond to those used in Figure 1.4.3-3.) On each analog input pin, a double switch, which is controlled by a single digital signal, will select either the test input or the normal input, but never both or neither. The buffered output of the double switch may be selected as the analog test output through a single switch, requiring a single digital control signal. Output pin switching nearly mirrors input pin switching. Here, the analog test output switch, used to observe the core analog circuit output, precedes the double switch and buffer combination controlling the analog output pin. Note that a buffer after the mux prevents mux resistance from affecting measurement or normal circuit operation.

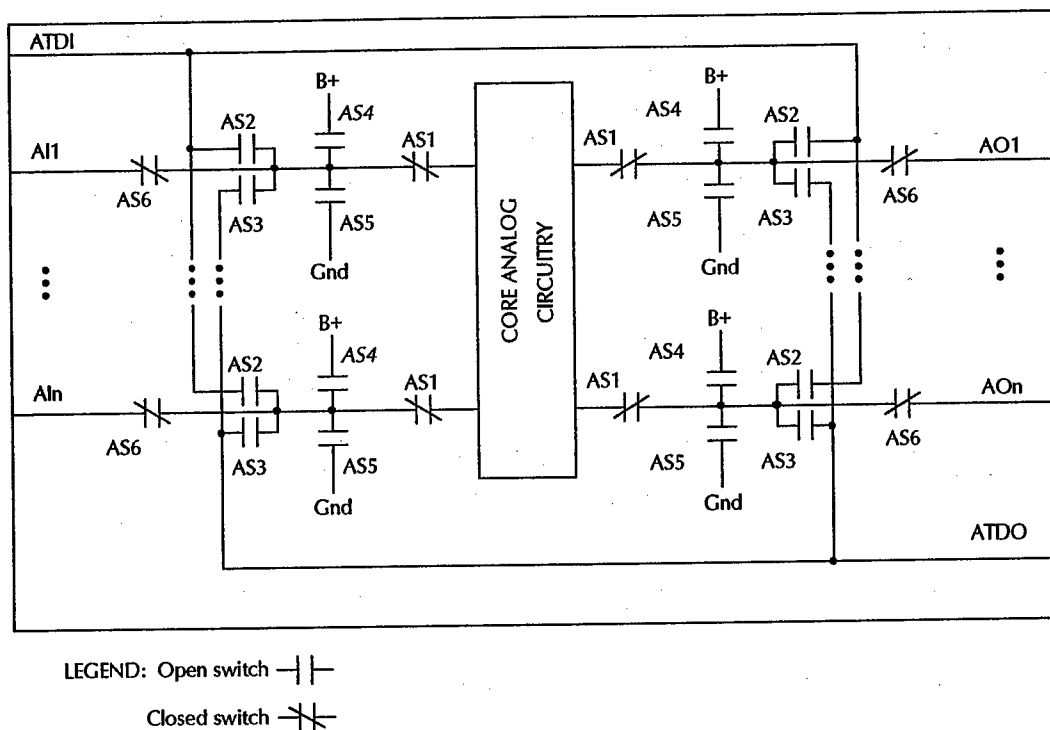


Figure 1.4.3-3. Six-Switch Implementation of Switching Network

A more complex network, shown in Figure 1.4.3-3, requires 6 switches with separate control signals and does not include the buffers. Table 1.4.3 describes the function of each switch.

Switches AS4 and AS5 function somewhat more like IEEE 1149.1 in that they provide individual test control values for each signal simultaneously. The benefit of this extra control must balance or, better, outweigh its overhead. Switches AS2 and AS3 essentially function the same in both example implementations. So do AS1 and AS6. However, in the minimum network, AS1 is used only on inputs and AS6 only on outputs. Including both creates a generic circuit, usable for inputs or outputs.

The final function of the network and also the gates to implement it and the digital logic to control it must be selected by the P1149.4 working group. The type and connection of muxes, switches or transmission gates must be resolved. Current inclination is toward power field effect transistors (FETs), because they are feasible and should lessen impact on circuit performance. Mux-demux trees may be preferable to the parallel-connect method used inside devices, if frequency response is important. Also, mux-demux trees would require only n control signals to select 1 of 2^n for test stimulus or response.

The type of network, as explained, will affect the number of digital control signals needed. The standard probably should define where all the digital controls for switches on the pins will be located. They may be controlled by IEEE 1149.1, but the instructions to access the scan registers are undefined. None, one or all control signals could be contained in the boundary scan register. Controls not in the boundary register could be grouped in one or

more other registers, possibly by function. While such trivial decisions could be left to the designer, advance specification creates consistency.

Table 1.4.3. Function of Six-Switch Network.

Analog Switch Number	Connection Function	Signal Connected
AS6 AS1	Observe from Observe from	analog input pin core analog circuitry output
AS3	Observe to	analog test output
AS2	Control from	analog test input
AS4	Control from	positive bias voltage
AS5	Control from	ground
AS1 AS6	Control to Control to	core analog circuitry input analog output pin

Optionally, a similar switching network could be used to surround analog macros inside an IC, analogous to the internal scan function of 1149.1. The network could be simpler. It should help test ICs during their manufacturing tests. Additionally, it may simplify board functional test, by adding critical observation or control of selected internal signals. Different signals could be selected as appropriate for any given section of the test. The standard would define limited requirements for this optional macro testing, as was done by IEEE 1149.1 for internal scan.

ICT.

Although this logic will function like ICT, it is actually built-in test (BIT), because it is contained within the unit under test (UUT). The proposed BIT logic provides one analog test input and measures one analog test output. An early framework, presented by IMP, implements BIT on each signal. Each input pin BIT selects control from one of two reference voltages. Each output pin BIT performs a 1-bit A/D conversion using a programmable bias voltage. By selecting a series of bias voltages, the voltage level of each output can be identified. Parallel stimulus application and response measurement is an advantage, but the overhead may be too high and testing is limited to DC voltages. A global or shared BIT resource can be more flexible and use less overhead. ICT as defined in the strawman is shown in Figure 1.4.3-4. It is an op-amp with its positive input tied to ground, a programmable current source on its negative input supplying stimulus to the UUT, and a digital/analog converter (DAC) to measure voltage on its output. With it the user could measure the following values:

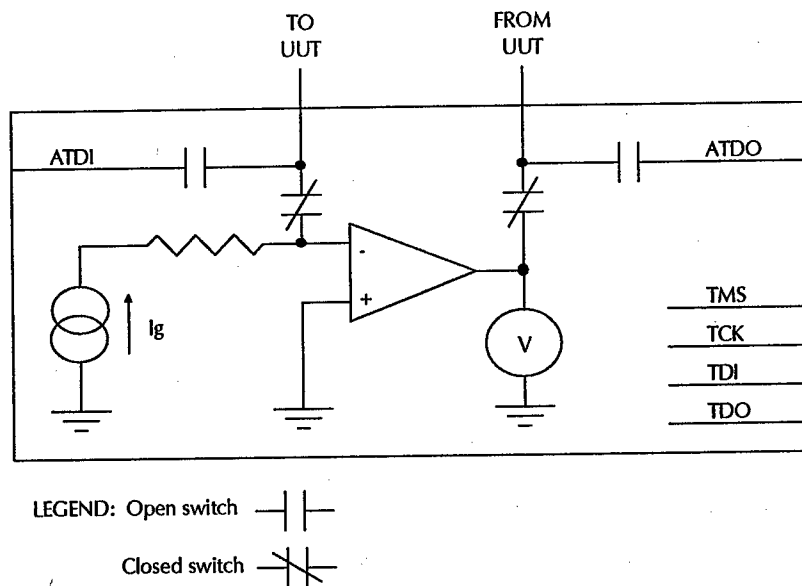


Figure 1.4.3-4. Global BIT Resource as Defined in ICT Strawman

Resistance	with programmed current set to necessary value
DC Voltage	with the programmed current set to zero
AC Voltage	with a measurement filter
Capacitance or Inductance	with an AC source and measurement filter
Frequency	with pair of measurement filters yielding varying degrees of accuracy

The use of a dedicated analog test IC to support BIT would permit integration of fairly sophisticated test capabilities, almost a test system on a chip. It could be a mini-ATE having one stimulus line with a programmable waveform generator and one response line with a programmable multi-meter. Control of the mini-ATE could be provided by IEEE 1149.1. The digital bus would control the selection of stimulus and type of response measurement, as well as scan out a digitized value of the data monitored.

Definition of ICT should not hold back releasing a proposed standard for approval. ICT could be a recommendation only, or, at most, some minimum function would be required.

Summary. For many reasons shared by digital test, primarily physical inaccessibility and circuit densities, a great need exists for standard mixed-signal test methods. The P1149.4 working group hopes to attract more participation from analog test engineers by

increasing awareness of the need and by holding meetings around more analog-oriented conferences. The strawman, discussed above, has stimulated vigorous discussion and research. With inputs and development effort from even more companies, the issues raised by that framework could be resolved this year. After that, release of a draft version would be forthcoming.

1.5. IEEE P1149.5/ TM-Bus.

1.5.1 Overview and Intended Use.

The IEEE P1149.5 Module Test and Maintenance Bus (MTM-Bus) proposes a standard, serial, backplane test & maintenance bus protocol for communicating test control and status information between modules within an electronic system. A module within IEEE P1149.5 may be defined by any logical subsystem partition: single board (most popular definition) or a cluster of multiple boards. IEEE P1149.5 is defined to allow the removal/replacement of modules without affecting MTM-Bus operation.

The MTM-Bus protocol supports singlecast, multicast, and broadcast addressing between a single bus master module and multiple slave modules (maximum of 250 Slaves). A module may be programmed by the Master to belong to any one of four separate multicast groups. Once addressed, both full- and half-duplex transfer of 16-bit serial data packets between the master and slave(s) is supported.

MTM-Bus communication is message oriented, as opposed to bit oriented (like serial scan buses). Each message consists of a series of 17-bit packets (16-bits data, 1-bit parity). The message includes the transfer of command/address information followed by the transfer of data packets. The bus also supports data flow control capabilities and Slave-to-Master interrupt capabilities.

Extensive bus error detection capabilities such as parity checking, illegal bus state sequencing, illegal command sequencing, bus stuck-at fault detection, signal collision detection, and data overrun error detection are key components of the bus protocol specification. Other forms of error detection, such as packet counting and address acknowledgment, are included as protocol options.

The MTM-Bus specification defines a minimum set of Core commands that execute regardless of the Slave's state and support minimum module control (initialization) and status reporting. Additional recommended commands are included for module initialization & self-test, module interconnect testing, and data transfer to on-module resources. A large user-defined command space is provided to support application-specific functions.

The MTM-Bus is intended to support system integration of testable modules, as well as provide a standard means for in-system fault management and diagnostic information flow between modules. It also provides a standard interface protocol for common module test resources (e.g., fault log, status registers, etc.).

IEEE P1149.5 is a protocol definition. It is intended to define a standard test and maintenance bus interface protocol sufficiently to allow:

- Flexibility to allow multiple system/backplane profile definitions. Profiles are necessary such that the MTM-Bus may support various backplane technology types, system sizes (number of modules), and varying degrees of fault tolerance.
- Bus (or link) level protocol and message level protocol interoperability. Link-level protocol interoperability ensures that two modules within the same profile will communicate with one another correctly. Message-level interoperability ensures a common system software architecture for interfacing between modules.
- Equal applicability/tailorability to commercial or military systems.

Commercial and military systems typically have different intended uses and therefore different sets of imposed requirements. Commercial systems are often simplified to ensure low-cost and decreased time-to-market. Military systems are often complex with many stringent requirements due to the rigorous environment that they must operate in. IEEE P1149.5 provides a minimum set of requirements and a large set of recommendations to allow the system/module designer with ultimate flexibility.

1.5.2. Current Status.

IEEE Standardization.

The IEEE P1149.5 specification successfully passed its first ballot in January 1993, with an 85% ballot return and 91% approval. There were five dissenting votes and comments from all voters. The P1149.5 working group activities are continuing with the negative ballot resolution process and a review of all of the comments received. The working group plans to submit the proposed standard for a second ballot in mid-1993 and receive final approval by the IEEE standards committee by the end of calendar year 1993.

The primary issues resulting from the first ballot comments include:

- Lack of definition of an IEEE P1149.5 to IEEE 1149.1 interface.
- Concern about bus throughput due to required minimum of four pause states following each packet transfer.
- Poor document organization; difficulty in discerning master-unique and slave-unique requirements/recommendations.

Each of these issues has been addressed and changes will be incorporated in the second ballot. The architecture details described in this evaluation anticipate the incorporation of these changes.

Relationship To Other Standards Efforts.

IEEE P1149.5 utilized the VHSIC Phase 2 TM-Bus interoperability standard 3.0 and the Joint Integrated Avionics Working Group (JIAWG) TM-Bus as the initial baseline for its standardization work. As the standard matured and improvements were made, the

interoperability between the IEEE P1149.5 MTM-Bus and the JIAWG TM-Bus grew apart. Following the F-22 (Advanced Tactical Fighter) and Commanche (Light Helicopter) contract awards, the JIAWG efforts were reduced from industry-wide participation to sole participation by the contracting companies. There was even some disagreement upon TM-Bus standardization among the F-22 and Commanche programs. As a result, the contracting companies formed a Commonalty Working Group (CWG) and launched a TM-Bus standardization effort to ensure interoperability between the two programs. IEEE representatives participated in the CWG meetings and had some influence in the CWG TM-Bus (aka JIAWG TM-Bus) definition; many of the issues had already been addressed by the IEEE P1149.5 working group. The CWG has decided to adopt the IEEE P1149.5 MTM-Bus protocol (to a great extent) and utilize the Society of Automotive Engineers (SAE) as the forum to standardize the JIAWG specific profiles. These profiles will be defined in SAE AS4765, Avionics TM-Bus Interoperability Standard, and will include bus electrical characteristics, bus duality options, and mastership arbitration/transfer options.

1.5.3. Interface/Number of Pins.

The IEEE P1149.5 MTM-Bus is defined as a four signal serial bus interface having a multidrop topology, as depicted in Figure 1.5.3. The four required signals are the clock-MCLK, master data-MMD, control-MCTL, and slave data-MSD. The MCLK signal is a synchronous reference for all bus transfers and is sourced by the system (typically independent from the master and slave signals). All other MTM-Bus signals are sourced on the logical rising edge of MCLK and are latched into the destination on the logical falling edge of MCLK. The MCTL signal and MMD signal are sourced by the bus master to the slave(s) and collectively control the current state of the MTM-Bus. MMD is also used to transfer data from the master to the slave during data transfer periods within the bus protocol. The MSD signal is sourced by the slave(s) to the bus master and is used to transfer data or signal an interrupt. An optional 5th signal, pause request-(MPR), may be included on MTM-Bus slaves to aid in controlling data flow across the bus. MTM-Bus master modules are required to implement the MPR signal so that it can communicate with any slave module.

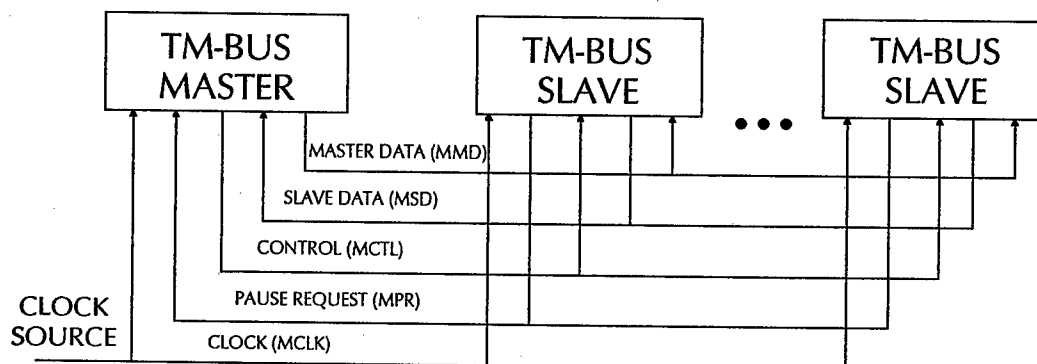


Figure 1.5.3. Architecture Interface

IEEE P1149.5 defines a minimal set of physical layer (electrical, timing) requirements for the signals that form the MTM-Bus to ensure module interoperability is maximized. This approach allows for the definition of multiple system/backplane profiles in the future. This is necessary due to the varying differences among system size, backplane technologies, fault tolerant requirements, etc.

The SAE AS-2C-2 TM-Bus task group is in the process of defining avionic profiles for IEEE P1149.5 that will support various system/backplane implementations. IEEE P1149.5 working group representatives are participating in the SAE effort. The working group is also investigating the definition of profiles through the IEEE.

1.5.4. Architecture.

The IEEE P1149.5 MTM-Bus architecture consists of a single bus master module and one or more slave modules (logical maximum of 250) residing on a single 4/5-wire multi-drop serial bus, as shown in Figure 1.5.4. The bus master controls all bus transfers to and from the slave(s). Each slave module is uniquely identified by a fixed 8-bit slave address. The slave address is typically implemented through dedicated backplane discretes, module resident DIP switches, or other programmable means. Messages may be transferred between the master and a single slave module (Singlecast), a designated group of slave modules (Multicast), or all modules (Broadcast). There are 255 potential slave addresses supported within P1149.5: 251 singlecast addresses (Hex 00-FA), 4 multicast addresses (Hex FC-FF), and 1 broadcast address (Hex FB). A slave module may be programmed by the master to respond to any one of the 4 multicast addresses. All slaves respond to the broadcast address. A slave having a matching module address will respond to the singlecast address.

Each bused signal of the MTM-Bus, excluding MCLK, is wired-OR such that a module may be removed or added without disrupting bus operation or yielding the bus unusable. The wired-OR nature of the signals also supports slave-to-master interrupt capability without requiring additional signals. It also provides for fault tolerant implementations in which multiple bus masters having a common mastership arbitration/transfer capability may coexist.

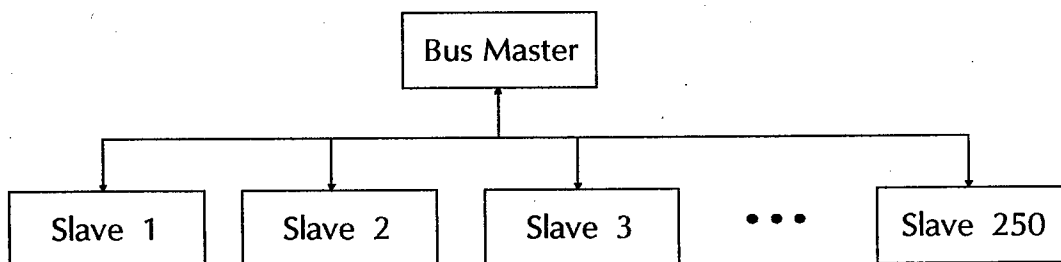


Figure 1.5.4. P1149.5 Test Bus Architecture

1.5.5. Bus Protocol.

The P1149.5 interface protocol is divided into three distinct layers as shown in Figure 1.5.5-1. These layers include the physical layer, the link layer, and the message layer. The physical layer describes the required input/output signals and their associated electrical requirements. The link layer describes the bus state transitions and the minimum requirements for status registers. The message layer describes the characteristics of packets that are transferred over the bus, the syntax for MTM-Bus messages, required/recommended functions supported via the bus, and error handling operations.

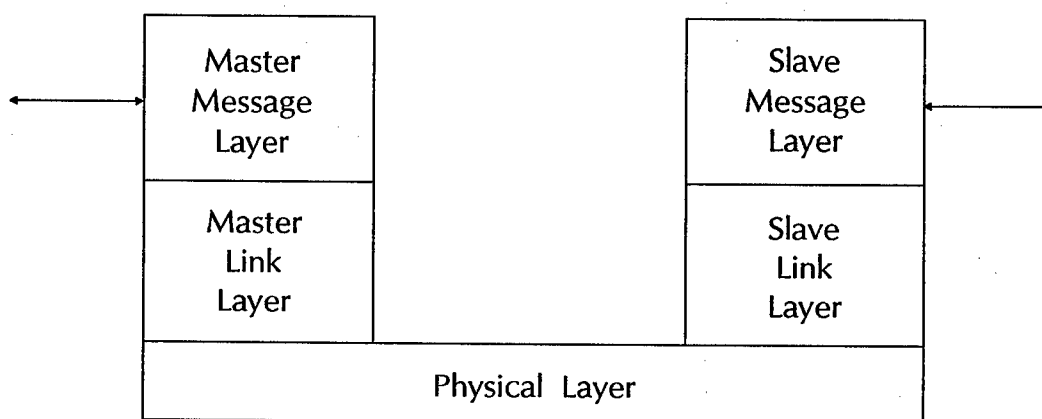


Figure 1.5.5-1. P1149.5 Protocol Layers

Physical Layer.

The IEEE P1149.5 physical layer defines the bus timing relationships and electrical interface characteristics for bus signals. P1149.5 specifies the worst-case timing relationships based upon system backplane parameters and module timing parameters. The selection of system performance times must adhere to the modules' capabilities. Although backplane electrical solutions are not addressed by IEEE P1149.5, recommendations are included to enhance interoperability. The exact electrical and timing specifications for bus signals are tailorable by the system engineer to meet his/her system needs. It is presumed that various physical layer profiles will be developed in the future to support a number of potential system implementations (e.g., high drive, low drive; large system, small system, etc.). The SAE AS-2C-2 TM-Bus task group is already working potential profiles for avionics equipment.

To ensure race-free operation, P1149.5 implements a dual edged clocking approach. All signals are sourced on the rising edge of MCLK and are latched on the falling edge of MCLK. The MMD, MCTL, and MSD signals are required to include a collision detection capability. Collision detection involves the module verifying that the bus actually assumes the state that is being sourced by the module. A mismatch between the source and the bus is the result of a bus collision, in which two modules are attempting to drive the signal to opposite states; one module will override. The module detecting the collision is required to release the signal and indicate an error condition (if appropriate). Signal collision

detection is integral to successful interrupt servicing on the MTM-Bus, as well as providing additional fault detection capability.

The MSD signal contains response data from the slave. A two MCLK delay is inserted between the bus transfer state and the data being sourced on MSD to allow for slave decode/response time. That is, a slave will begin to source MSD with bit 16 of a packet when the bus enters the transfer 14 (XFER14) state.

Link Layer.

The MTM-Bus link layer defines the lowest level bus protocol state diagrams and minimum requirements for MTM-Bus accessible registers.

Slave State Machine.

The slave link layer state diagram is shown in Figure 1.5.5-2. This state diagram defines the bus state sequences involved in the transfer of each 17-bit packet.

The IDLE state indicates that there is no message transfer occurring between the master and slave module(s). This is the second of two states required between each MTM-Bus message (i.e., bus protocol requires a minimum of two MCLK cycles while MCTL and MMD are released). The packet transfers between two IDLE states constitute an MTM-Bus message.

The transfer states (XFER16-XFER0) are used to force the transfer of a packet (17 bits) between the master and slave module(s). Packets are always transferred MSB (bit 16) first. The LSB of the packet is the packet parity bit and is transferred during the XFER0 state.

The pause states (PAUSE1, PAUSE2, PAUSE3, PAUSE) are states required between each packet transfer to provide a minimum delay such that slave modules can interpret and respond as necessary. A minimum of four pause states are required between packets. (Note: In the interest of increasing data throughput, the second ballot draft specification will change this requirement to a minimum of four pause states following the first packet transfer of the message and a minimum of one pause state on subsequent packet transfers). The master may insert additional pause states beyond the minimum required by remaining in the PAUSE state. The slave may request that the master insert additional pause states by asserting the MPR signal.

The IDLE1 state indicates the end of a message transfer between the master and slave. It also allows the slave time to process and respond to any post-message errors that may occur.

The ERROR state is a slave state used to ensure a consistent slave response to bus protocol error conditions. It provides the slave with a place to wait until the completion of the message when a state sequence error causes the slave and master link layers to become unsynchronized. Individually addressed slave modules generate an interrupt to the master during the ERROR state by asserting the MSD signal. This guarantees that error detection latency by the master is reduced to a minimum.

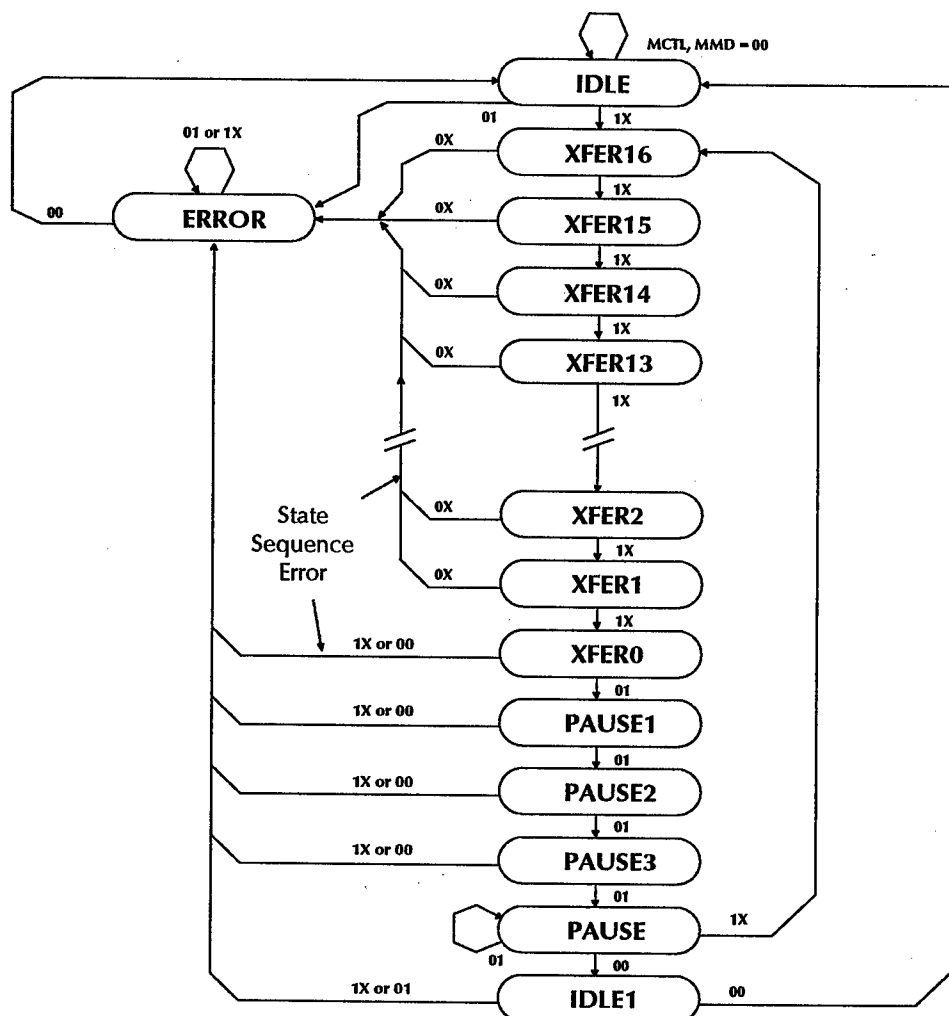


Figure 1.5.5-2. Link-Layer Bus States

MTM-Bus Registers.

The MTM-Bus requires a minimum of two status registers, and recommends a third, for indicating MTM-Bus slave status, bus error conditions, and module status information. Additional status registers may be implemented to meet the needs of the module.

The Slave Status Register provides key information to the master about the MTM-Bus slave including:

- the module's start-up built-in test results,
- the processing status for the previously received MTM-Bus message,
- the primary cause for a slave generated MTM-Bus interrupt,
- the currently selected multicast group for the slave, and
- the current interrupt capabilities for the slave.

The Bus Error Register provides specific causes for link and message layer error conditions which may have occurred during or immediately following the message. The potential bus errors include:

- Broadcast/multicast message not received properly,
- Slave data fault; i.e., a bus collision,
- Command resource unavailable,
- Incorrect packet count,
- Illegal port selected,
- Port transfer error,
- Command sequence error,
- Illegal command,
- Packet parity error,
- Data overrun error, and
- State sequence error.

The Module Status Register is an optional register that provides a module with a place to include:

- Specific causes for interrupts sourced by the module application (i.e., not necessarily related to the MTM-Bus), and
- Module-specific status information.

Message Layer.

The message layer defines the characteristics of packets which constitute an MTM-Bus message, the required syntax for MTM-Bus messages, and the functions which must or should be supported by the bus master and slave(s).

Packet Types.

There are five packet types possible within an MTM-Bus message, as indicated in Table 1.5.5-1.

Table 1.5.5-1. MTM-Bus Packet Types

Packet	Originator	Description
HEADER	Master	First packet in an MTM-Bus message
PACKET COUNT	Master	Optional; number of additional packets
ACKNOWLEDGE	Slave	Optional; Slave status information
DATA	Master or Slave	Binary Data
NULL	Master or Slave	Special Data packet

The HEADER packet is the first packet transferred within a message and contains the target slave module address, a command field and an acknowledge request bit. The slave module address identifies the target slave(s) the master wishes to include within the MTM-Bus message. The command field defines the remainder of the message sequence and the actions that will take place. The master may request a slave to transmit an ACKNOWLEDGE packet as the first packet returned by setting the acknowledge request bit in the HEADER packet.

The PACKET COUNT packet, if included within a message, is transferred as the first packet following the HEADER. The PACKET COUNT packet contains a value indicating the number of additional packets that will be transferred within the message. If the value is zero, then the number of additional packets is undefined. The PACKET COUNT packet is optional and is intended to be utilized with large MTM-Bus messages to aid in error detection.

The ACKNOWLEDGE packet, if included within a message, is the first packet transferred by the slave. If requested in the HEADER packet, the ACKNOWLEDGE packet is transferred coincident with the PACKET COUNT packet from the master. Some MTM-Bus commands (Read Status, Contend, ...) require the ACKNOWLEDGE packet to be transferred regardless of the state of the acknowledge request bit in the HEADER packet. The ACKNOWLEDGE packet contains the slave's 8-bit module address and the contents of the Slave Status Register.

DATA packets contain binary data as defined by the MTM-Bus command in the HEADER and are transmitted as defined by the message sequence for that command. Binary data of less than 16 bits in length is LSB justified within the packet with a recommendation that the unused bits be set to zero. Binary data of greater than 16-bits (i.e., spans multiple DATA packets) is transmitted in little ENDIAN format, with the least significant 16-bits in the first packet, the next 16 least significant 16-bits in the second packet, etc.

NULL packets are a unique form of DATA packets. The NULL packet consists of all zeros with good parity. NULL packets are transferred by the sending module (Master or

Slave), following the HEADER packet, when the receiving module does not expect a valid DATA packet.

Message Format.

The standard MTM-Bus message format is shown in Figure 1.5.5-3. Each message consists of a HEADER packet as a minimum. The HEADER packet may be optionally followed by a ACKNOWLEDGE/PACKET COUNT packet pair if desired. The remainder of the message consists of DATA packets transferred in full-duplex or half-duplex mode between the master and slave.

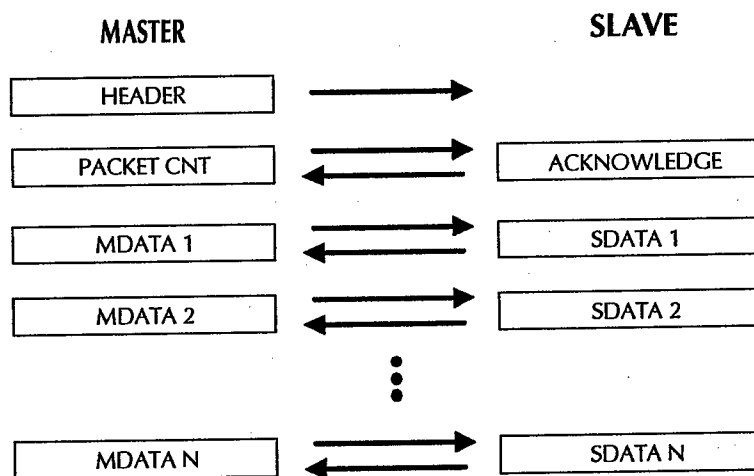


Figure 1.5.5-3. Generic P1149.5 Message Format

P1149.5 Commands.

IEEE P1149.5 defines 12 required command codes and 17 recommended command codes as shown in Table 1.5.5-2. MTM-Bus commands are divided into five classes: the Core class, Data Transfer class, Module Initialization and Self-Test class, Module I/O Control and Test class, and User-Defined class. The Core class contains all of the required commands. These commands are required to execute regardless of the current state of the module.

For each command in the table, the message type identifies the general message format to be followed for that command. The possible message types are simple, send, receive, and send/receive. Simple messages consists of a HEADER packet and optional ACKNOWLEDGE/PACKET COUNT packet pair only. These commands typically perform a well defined function such as enable interrupts, reset the module, etc., that does not require additional data. Send and receive type messages are considered half-duplex; expected data only flows in one direction, to the slave or to the master. NULL packets are transmitted by the receiving module (master or slave) opposite to the expected data. Send messages are those messages in which the slave sends data to the master in response to the command. For example, reading slave status is a send type message. The master transfers NULL packets to the slave while the slave transfers the contents of internal

registers to the master. Receive messages are those message in which the slave receives data from the master. The slave transfers NULL packets to the master while the master transfers a series of expected data packet to the slave. Send/Receive messages are considered full-duplex; expected data flows in both directions, from master to slave and from slave to master.

For a given message, there may be a fixed number of packets of delay between the slave receiving a command/data and the slave transmitting the response packets. This delay is known as the "packet latency". Packet latency is specified for all required commands and many of the recommended commands. The packet latency is defined by the module designer for any user-defined commands or data transfer ports.

Data Transfer Ports.

The MTM-Bus protocol defines a standard data transfer message class. This class of commands supports access to a maximum of 65536 user-defined data transfer ports. Each data transfer port is individually addressable via the MTM-Bus. Various types of access to the port is provided such as read, write, read/write, read indirect, write indirect, etc. IEEE P1149.5 provides recommended data transfer port definitions for some typical module resources such as status registers, module fault log, IEEE 1149.1 bus, etc. These definitions standardize the message format for accessing the port.

Interrupts.

Unlike other serial scan-type buses, the MTM-Bus supports an interrupt capability without the need for additional bus signals. This is accomplished by sharing the function with normal slave data transfer on the MSD signal. The MSD signal contains slave data during the bus transfer states (XFER16-XFER0) and, if asserted, indicates a slave interrupt condition during non-transfer bus states (Idle, Pause, ERROR). Interrupts may be signaled due to a bus error condition detected by the slave or a request from the module application. The Bus Error Register and Module Status register identify the interrupt causes.

Interrupt Servicing.

To service an MTM-Bus interrupt, the master module issues the Contend For Bus message. During the contend, all interrupting slave modules source their slave address onto the MSD signal, one bit at a time. Every bit sourced is compared with the actual bus state using the required collision detection logic. A miscompare, or bus collision, results in the slave detecting the collision to discontinue the contend by releasing MSD. Since all packets are transferred MSB first, the resulting packet received by that master will identify the interrupting slave with the highest module address. After servicing this interrupt, the master re-issues the Contend For Bus message, and continues servicing each interrupting module in the same manner.

Table 1.5.5-2. MTM -Bus Command Codes

Command Class	Command Codes	Command	Message Type	Required?
Core	0000000	Read Status	Send	Required
	0000001	Initialize Module	Simple*	Required
	0000010	Reset Slave Interface	Simple	Required
	0000011	Contend For Bus	Simple	Required
	00001xx	Multicast Select	Simple	Required
	0001000	Enable Idle Interrupts	Simple	Required
	0001001	Enable Pause Interrupts	Simple	Required
	0001010	Disable Idle Interrupts	Simple	Required
	0001011	Disable Pause Interrupts	Simple	Required
	0001100	Enable Module Control	Simple	Required
	0001101	Data Echo Command	Send/Receive	Required
	0001110	Reserved		Required
	to 0011111			
Data Transfer	1111111	Illegal Command	Simple	Required
	0100000	Read Data	Send/Receive	Recommended
	0100001	Write Data	Receive	Recommended
	0100010	Read/Write Data	Send/Receive	Recommended
	0100011	Read Data Indirect	Send/Receive	Recommended
	0100100	Write Data Indirect	Receive	Recommended
	0100101	Write Data Indirect/Update	Receive	Recommended
	*0100110	Read Data Indirect/Update	Send/Receive	Recommended
Module Init. & Self-test	0100111	Reserved		Required
	0101000	Reset Module with SBIT	Simple*	Recommended
	0101001	Reset Module without SBIT	Simple*	Recommended
	0101010	Module IBIT	Simple*	Recommended
	0101011	Reserved		Required
Module I/O Control & Test	to 0101111			
	0110000	Disable Module I/O	Simple*	Recommended
	0110001	Enable Module I/O	Simple*	Recommended
	0110010	Force Module I/O	Receive*	Recommended
	0110011	Sample Module - No Change	Send*	Recommended
	0110100	Sample Module - Don't Care	Send*	Recommended
	0110101	Sample Module with Force	Send/Receive	Recommended
	0110110	Release Module I/O	*	Recommended
	0110111	Reserved	Simple	Required
	to 1001111			
User	1010000	User-defined Commands		
	to 1111110			

* Indicates that an Enable Module Control (EMC) command must precede this command

1.5.6. Bus Timing.

To ensure race-free operation, the P1149.5 MTM-Bus uses a dual edge clocking approach. The system designer is allowed the flexibility to define the MCLK high and low time in order to maximize performance.

All MTM-Bus timing is based upon system and module timing parameters. These parameters are typically defined by the target system and will therefore often be defined by a system specific timing profile. Each MTM-Bus module must document its module timing parameters. The MTM-Bus timing relationships can be summarized as follows:

$$t1 \geq T1$$

$$t2 \geq T2$$

$$tc \geq Tc,$$

$$t1 \geq Tco + tp + Ts + tcs$$

$$t2 \geq Th + tcs$$

where:

t1 = MCLK high time

t2 = MCLK low time

tHL = MCLK transition time from valid logic 1 to valid logic 0

tLH = MCLK transition time from valid logic 0 to valid logic 1

tc = MCLK cycle time ($tc = tLH + t1 + tHL + t2$)

T1 = minimum operable t1 for the module

T2 = minimum operable t2 for the module

Tc = minimum operable tc for the module (time from the start of logic 1 to the start of next logic 1)

Tco = module clock-to-output time

tp = backplane propagation and settling time

Ts = module setup time

tcs = clock skew as measured between the module driving and the module receiving

Th = module hold time

Note that Tx is a module specific timing parameter and ty is a system specific timing parameter.

1.6. IEEE 488 (GPIB).

1.6.1. Overview and Intended use.

The IEEE 488 standard digital interface for programmable instrumentation has been the prominent system interconnect and communications bus used in engineering and scientific workstations since its adoption in 1975. The IEEE 488, or as it is commonly called, general-purpose interface bus (GPIB), was created to establish order amongst the numerous instrumentation interfaces in use during the mid-'70s. The standard established the electrical, mechanical, and functional characteristics of the bus, as well as the connector. The bus is designed to be used in limited distance applications, allows asynchronous communications, and supports a wide range of data rates.

The beauty of the IEEE 488 is its capability to permit independently manufactured and self-contained instruments, with a wide range of capability, to be interconnected, communicate, and form a single functional system. This interconnectivity has minimum restrictions on the performance characteristics of the connected instruments, and enables functional systems to be created with low cost 'off-the-shelf' instruments.

1.6.2. Current Status.

This general-purpose interface bus is currently in use throughout industry and government. In 1987, the capability of the IEEE 488 bus was revised, expanded, and renamed IEEE 488.1. This renaming was done to allow the expanded bus to be known as IEEE 488.2. The IEEE 488.2 bus supports the original capabilities and design characteristics of the IEEE 488.1 bus with enhancements in the areas of commands, code data formats, and protocol. Though the IEEE 488 bus (IEEE 488.1) has been around for some 20 years, it is still the most widely used test and measurement bus in the market place.

Testing Application.

The IEEE 488's capability to control, monitor and provide common cabling to instruments has allowed it to remain a popular test bus for use in industry and government applications. Its precise command language allows it to setup and trigger device events, and control the capture of instrument data for analysis by microprocessors. Several embedded systems have used the IEEE 488 bus as a means to communicate integration and test information. The YF-22A Mission Display Processor and the F-16 Modular Mission Computer systems use the IEEE 488 to connect to the user console to control embedded emulation logic. Via the IEEE 488, operations such as upload/download, read/write memory, run, stop, single-step, and examine or modify registers can be performed.

Table 1.6.3. IEEE 488 Bus Signal Lines

DATA BUS	TRANSFER CONTROL	INTERFACE MANAGEMENT
DIO1	DAV	IFC
DIO2	NRFD	ATN
DIO3	NDAC	SRQ
DIO4		REN
DIO5		EOI
DIO6		
DIO7		
DIO8		

1.6.3. Interface / Number of Pins.

The IEEE 488 standard specifies a single type of 24-pin connector in order to minimize cable requirements. The bus structure is organized into three sets of signal lines; data bus, data byte transfer control, and general interface management. The identification of each signal within the three sets is shown in Table 1.6.3. The eight data bus lines are used for parallel data communications. The transfer control bus lines provide a three-signal handshake scheme to transfer each data byte of information across the bus. The five interface management bus lines are used to provide an orderly flow of data from one bus device to another.

The functions supported by the standard are identified in section 1.6.6. Using these functions, the designer defines the capability of the device. Each device capability requires the use of one or more signals from the above three sets of signal lines. Based on the minimum capability of a device, the minimum set of signal lines required to be compatible with the IEEE 488 bus consists of the following:

- DIO 1-7
- DAV, NRFD, NDAC
- IFC and ATN (unnecessary in systems without a controller)

1.6.4. Architecture.

The basic IEEE 488 system consists of five components; controller, 488 bus, source and measuring instruments, unit under test interface (e.g. fixture), and the unit under test, and the IEEE 488 bus. Figure 1.6.4-1 illustrates this basic setup. Within a basic IEEE 488 system, the controller is a computer which manages and directs all communications over the bus network. Instruments which source signals and measure parameters are required for test measurement system operations. Test fixtures typically interface the source and measurement instruments to the unit to be tested.

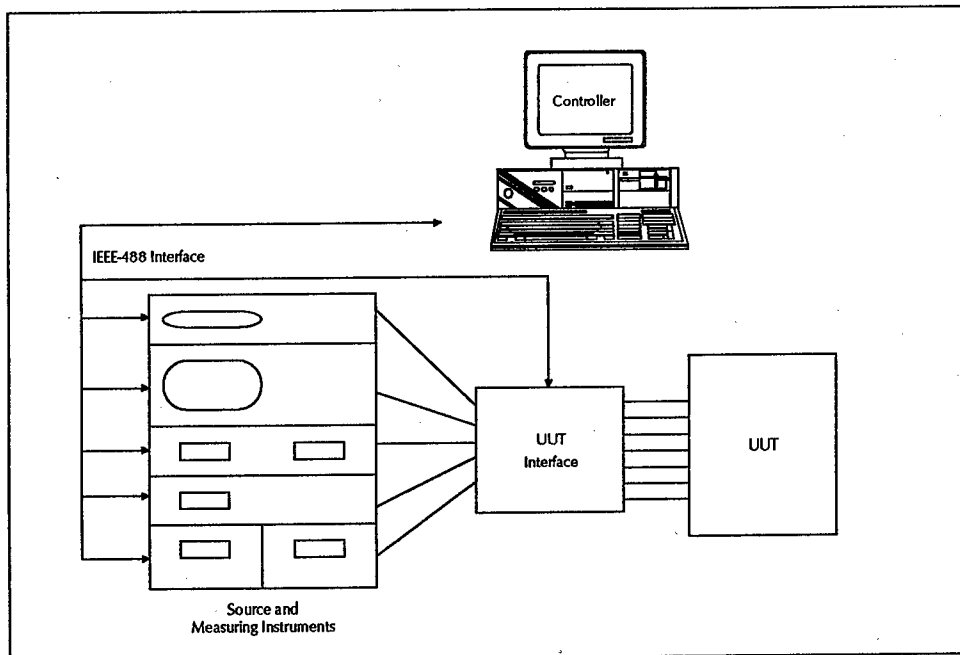


Figure 1.6.4-1. Basic 488 System Architecture

Test and measurement systems using the IEEE 488 bus can be configured in a number of ways, one of which is represented by Figure 1.6.4-2. The IEEE 488 bus can be either in a star or daisy chain configuration.

The IEEE 488 bus supports a total of 15 devices. This total includes the controller, all source and measuring instruments, and the interface to the unit to be tested. The IEEE 488 bus is designed for short data path applications with the total transmission path length not exceeding 20 meters. The bus exhibits relatively low electrical noise during operations. Bus drivers and receivers are based on TTL technology.

The IEEE 488.2 standard, is an enhancement to the original standard and defines additional, higher level capabilities. Among the changes are:

- more precise message exchange protocols,
- a defined process for handling multiple and erroneous messages,
- predefined commands which apply to all instruments,
- defined formats for handling integers, arrays, characters, etc., and
- a defined format for handling reporting of errors.

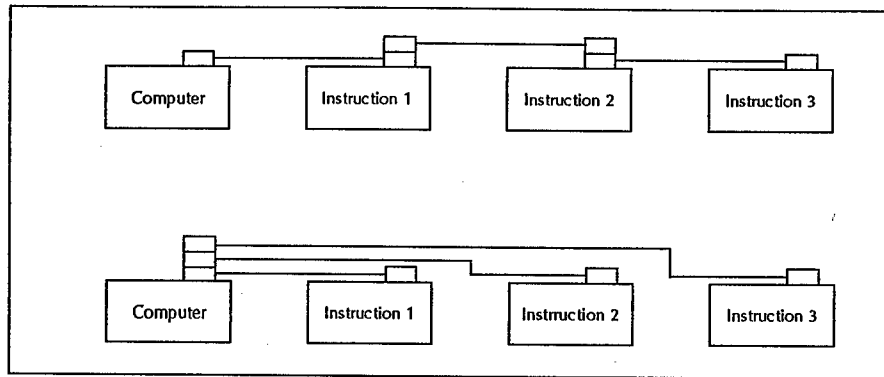


Figure 1.6.4-2. Typical IEEE 488 System Configuration

1.6.5. Protocol.

The IEEE 488 bus architecture introduced the concept of device listeners, device talkers, and device controllers for establishing an orderly flow of information across the interface. Bus protocol makes use of these three types of device capability and the signal lines identified above. Message bytes are carried on the DIO 1-7 signal lines in a bit parallel byte-serial form. Message flow is generally in both directions and can be asynchronous. Two types of messages can be transferred over the bus; uniline and multiline. Uniline messages are messages transmitted over a single bus line and are one of the interface management lines. Several uniline messages can be sent concurrently over the bus. Multiline messages consist of several signal lines grouped together in an exclusive set of signals. Only one multiline message can be sent at one time.

Three signal lines (DAV, NRFD, and NDAC) control the data bytes across the bus from the device acting as a controller or talker to one or more devices acting as listeners. Signal DAV (for data valid) is used to indicate whether or not data is available and valid on the bus. Signal NRFD (for not ready for data) identifies whether the device on the bus is ready to accept incoming data. Signal NDAC (for not data accepted) is used to signal the condition of acceptance of data by the device.

Five interface signal lines are used to manage an orderly flow of information across the interface:

- ATN (attention) is used (by a controller) to specify how data on the DIO signal lines are to be interpreted and which devices must respond to the data
- IFC (interface clear) is used (by a controller) to place the interface system, portions of which are contained in all interconnected devices, in a known state
- SRQ (service request) is used by a device to indicate the need for attention and to request an interruption of the current sequence of events
- REN (remote enable) is used (by a controller) in conjunction with other messages to select between two alternate sources of device programming data, either front panel or 488 control

- EOI (end or identify) is used (by a talker) to indicate the end of a multiple byte transfer sequence or, in conjunction with ATN (by a controller), to execute a polling sequence

A device on the IEEE 488 bus has an address which is usually set via switches on its rear panel. Devices may have any or all of the following capabilities: controller, talker, and listener. Only one controller can be allowed within the system at a time. Table 1.6.5-1 identifies messages which can be received and transmitted over the bus.

As mentioned in section 1.6.2, the IEEE 488.2 bus supports the original capabilities and design characteristics of the IEEE 488.1 bus, with enhancements. Devices compliant with the 488.2 bus must support a minimum set of capabilities. As a minimum, devices must be both talkers and listeners. Several message formats are defined and include, 7-bit ASCII code for text messages, and binary floating-point numbers using the IEEE-754-1985 standard format. An enhanced command set is defined to provide control of devices connected to the bus. Some commands are standard while others are optional. Table 1.6.5-2 list the commands and their functional group.

1.6.6. Functions supported.

As illustrated in Figure 1.6.6, a device connected to the IEEE 488 bus can be defined as processing five distinct functional areas; the interface bus, drivers and receivers, message coding, interface functions, and device dependent functions. Device dependent functions are beyond the scope of this report.

Table 1.6.5-1. IEEE 488.1 Messages

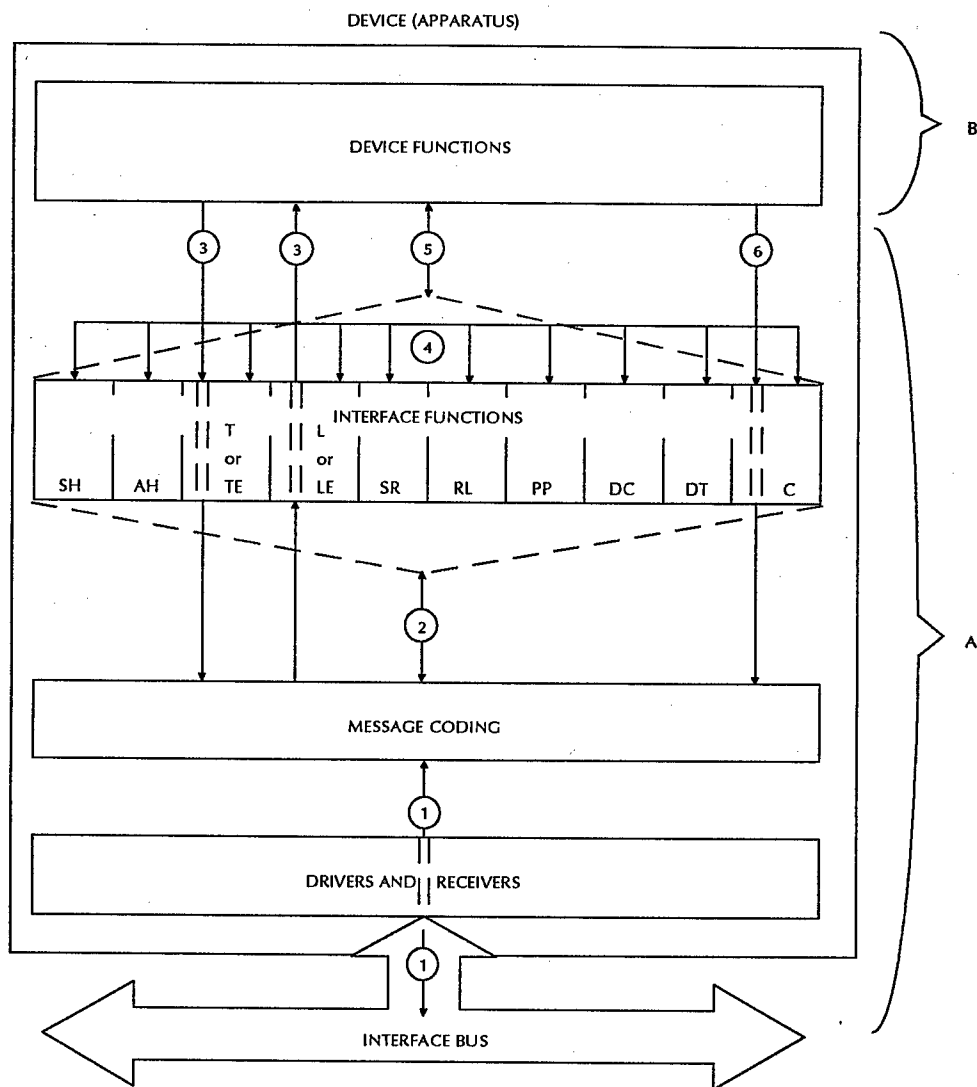
Mnemonic	Message
ATN	attention
DAB	data byte
DAC	data accepted
DAV	data valid
DCL	device clear
END	end
GET	group execute trigger
GTL	go to local
IDY	identify
IFC	interface clear
LLO	local lockout
MLA	my listen address
MSA	my secondary address
MTA	my talk address
OSA	other secondary address
OTA	other talk address
PCG	primary command group
PPC	parallel poll configure
PPD	parallel poll disable
PPE	parallel poll enable
PPRn	parallel poll response
PPU	parallel poll unconfigure
REN	remote enable
RFD	ready for data
RQS	request service
SDC	selected device clear
SPD	serial poll disable
SPE	serial poll enable
SRQ	service request
STB	status byte
TCT	take control
UNL	unlisten

Table 1.6.5-2. IEEE 488.2 Messages

Command Group	Cmd Mnemonic	Description	Required / Optional
Auto Configure	*AAD	Assign address	Optional
	*DLF	Disable Listener Function	Optional
System Data	*IDN	Identification query	Required
	*OPT	Option ID query	Optional
	*PUD	Protected user data	Optional
	*PUD?	Protected user data query	Optional
	*RDT	Resource description transfer	Optional
Internal Operation	*CAL?	Calibration query	Optional
	*LRN?	Learn device-setup query	Optional
	*RST	Reset	Required
	*TST	Self-test query	Required
Synchronization	*OPC	Operation complete	Required
	*OPC?	Operation complete query	Required
	*WAI	Wait to complete	Required
Macro commands	*DMC	Define macro	Optional
	*EMC	Enable macro	Optional
	*EMC?	Enable macro query	Optional
	*GMC?	Get macro contents query	Optional
	*LMC?	Learn macro query	Optional
	*PMC?	Purge macros	Optional
Parallel Poll	*IST?	Individual status query	Required&
	*PRE	Parallel poll enable register	Required&
	*PRE	Parallel poll enable query	Required&
Status and event	*CLS	Clear status	Required
	*ESE	Event status enable	Required
	*ESE?	Event status enable query	Required
	*ESR?	Event status register query	Required
	*PSC	Power on status clear	Optional
	*PSC?	Power on status clear query	Required
	*SRE	Service request enable	Required
	*SRE?	Service request enable query	Required
	*STB?	Read status byte query	Required
Device trigger	*DDT	Define device trigger	Optional
	*DDT?	Define device trigger query	Optional
	*TRG	Trigger	Required\$
Controller	*PCB	Pass control back	Required#
Stored settings	*RCL	Recall instrument state	Optional
	*SAV	Save instrument state	Optional

Where:

Required&	Required with parallel-poll capability
Required\$	Required with device-trigger capability
Required#	Required with system-controller capability



- A = Capability defined by this standard
- B = Capability defined by the designer
- 1 = Interface bus signal lines
- 2 = Remote interface messages to and from interface functions
- 3 = Device dependent messages to and from device functions
- 4 = State linkages between interface functions
- 5 = Local messages between device functions and interface functions
(messages to interface functions are defined, messages from interface functions exist according to the designer's choice)
- 6 = Remote interface messages sent by device functions within a controller

Figure 1.6.6. General IEEE 488 Functional Areas

The following interface functions are supported by the standard.

- Source Handshake (SH) controls the initiation of, and termination of multiline messages.

- Acceptor Handshake (AH) along with the SH function, controls proper reception of multiline messages.
- Talker or Extended Talker (T or TE) allows devices to send device dependent data over the interface. Address Extension is provided under the TE option.
- Listener or Extended Listener (L or EL) allows devices to receive device dependent data over the interface. Address Extension is provided under the LE option.
- Service Request (SR) allows devices to request service asynchronously from the bus controller.
- Remote Local (RL) allows devices to select between two sources of incoming data; either from the local front panel controls, or from the bus interface.
- Parallel Poll (PP) allows devices to present a parallel poll response to the controller without being addressed to talk.
- Device Clear (DC) allows devices to be cleared (initialized) either independently or as part of a group of devices.
- Device Trigger (DT) allows devices to be started either independently or as part of a group of devices.
- Controller (C) allows devices to send device addresses, address commands, and universal commands to other devices over the bus interface.

1.6.7. Speed.

The interface bus will operate at distances up to 20m, at a maximum of 250,000 bytes per second, with an equivalent standard load for each 2m of cable using 48mA open collector drivers. The interface bus will also operate at distances up to 20m at a maximum of 500,000 bytes per second, with an equivalent standard load for each 2m of cable using 48mA three-state drivers. To achieve maximum possible data transfer rate (nominally up to 1,000,000 bytes per second) within a system, the following is required:

1. devices should use a minimum signal settling time for multiline messages of 350ns.
2. devices should use 48mA three-state drivers.
3. device capacitance on each lead (REN and IFC excepted) should be less than 50pF per device. The system total device capacitance should be no more than 50pF for each equivalent resistive load in the system.
4. devices in the system should be powered on.
5. interconnecting cable links should be as short as possible up to a maximum of 15m total length per system with at least one equivalent load for each meter of cable.

NOTE: Anytime a device following condition (1) is placed in a system, even if higher speed operation is not intended, there may be data transfer errors if conditions (2) through (5) are not met for that system.

1.7. MIL. STD. 1553B/ 1773.

1.7.1. Overview and Intended use.

The MIL-STD-1553B data bus provides a communication path between processing resources and existing off-the-shelf equipment. The bus uses serial digital pulse code modulation and supports word and message validation, single/multi-topologies, and three types of terminals; bus controller, bus monitor, and remote terminal (see Figure 1.7.1).

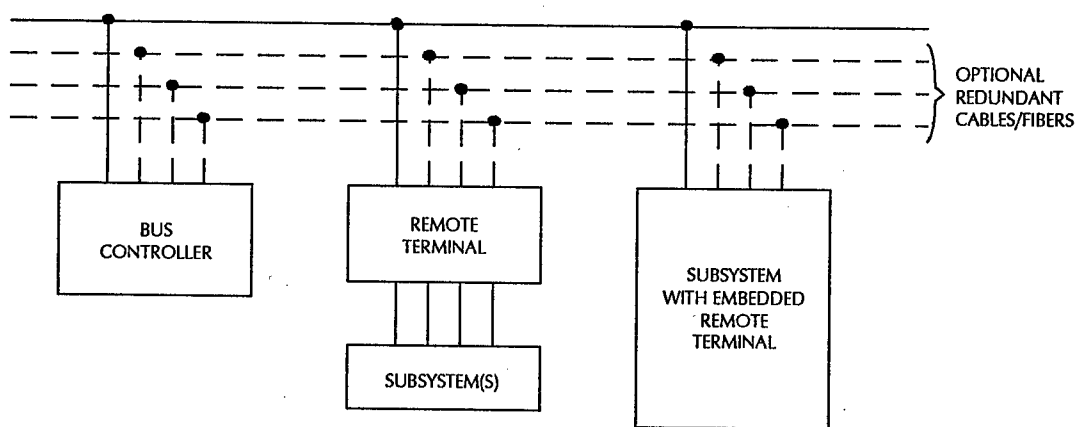


Figure 1.7.1. Typical Multiplex Data Bus Terminal Connectivity

The 1553 Bus is a serial data bus based on message transmissions. Currently there are 10 types of messages with associated message formats. Each message consists of control words, status words, and data words. Up to 31 terminals can be placed on the data bus, and each terminal can service up to 30 subsystems.

The MIL-STD-1773 was adopted in 1988 to provide fiber optic transmission within a 1553-type medium. Bus connectivity is similar to that of Figure 1.7.1.

1.7.2. Current Status.

MIL-STD-1553 is a mature standard which is being used by the military and industry. Originally adopted by the Tri-Service and other industry elements in 1973, it was revised in 1978 and became MIL-STD-1553B. It has achieved wide success in numerous applications in military programs for its ability to support integration of avionic subsystems. The standard has allowed interchangeability of equipment from numerous suppliers due to its flexibility to support multiple supplier designs.

MIL-STD-1773 provides for the use of fiber optics in implementing a 1553-type network. This standard preserves the 1553 multiplex bus techniques while it provides for the use of fiber optics to take advantage of the wide bandwidth characteristics of the fiber. The

current 1773 specification is identical to the 1553 specification except where fibers characteristics are used instead of wire characteristics.

Testing Application.

Current avionic platforms have used the 1553 to provide status information to assist in mission decisions. Embedded condition monitoring applications for the V-22 tiltrotor aircraft subsystem uses the 1553 bus to obtain vibration, structural, and engine monitoring information to assist in identification of potential vehicle failures and to schedule maintenance operations based on component usage and not component failure. Via the 1553, condition monitoring information can be obtained from engine diagnostic applications, monitoring of rotor and gear box vibrations, and structural component usage at speeds which allow decisions to be made automatically by the system, or by the pilot, as to the flight worthiness of the vehicle.

The use of a 1773 bus is being studied to determine its applicability in systems similar to the V-22. The light weight of the 1773 fiber-optics bus would go a long way to reducing the weight penalty experienced by aircraft contractors implementing condition monitoring subsystems, while at the same time maintaining or surpassing bus throughput obtained by 1553 buses. The use of a 1773 fiber-optics bus would allow more extensive monitoring of vehicle flight-critical and safety-critical subsystems due to its light weight.

1.7.3. Interface / Number of Pins.

The interface requirements associated with using the 1553 within system designs are listed in Table 1.7.3. Minor variations from the specified cable characteristics has been shown to still provide adequate system performance.

Current 1773 interface requirements support those of the 1553 specification; identified by the use of an asterisk (*). Transmission line and cable coupling parameters are currently not defined.

Table 1.7.3. 1553B/1773 Data bus/Coupling Requirements

Parameter	MIL-STD-1553B
General	
Applications	DoD Avionics (*)
Data Rate	1 MHz (*)
Word length	20 bits (*)
Numner of data bits/word	16 (*)
Transmission technique	half-duplex (*)
Operation	Asynchronous (*)
Encoding	Manchester II biphase (*)
Transmission line	
Cable type	Twisted-shieled pair (fiber optic cable for 1773)
Capacitance (wire to wire)	30 pF/ft, maximum
twist	Four per foot (0.33/in), minimum
Characteristic impedance (z_0)	70 to 85 ohms at 1.0 MHz
Attenuation	1.5 dB/100 ft at 1.0 MHz, maximum
Length of main bus	Not specified
Termination	Two ends terminated in resistors equal to $z_0 \pm 2\%$
Shielding	75% coverage minimum
Cable coupling	
stub definition	Short stub < 1ft. , Long stub > 1 to 20 feet
Coupler requirement	Direct coupled--short stub; transformer coupler--long stub
Coupler transformer	
Turns ratio	1:1.41
Input impedance	3,000 ohms, min. (75 kHz to 1.0 MHz)
Droop	20% maximum (250-kHz)
Overshoot and ringing	$\pm 1.0V$ peak (250-kHz square wave with 100-ns max. rise and fall time)
Common mode rejection	45.0 dB at 1.0 MHz
Fault protection	Resistor in series with each connection equal to $(0.75z_0) \pm 2.0\%$ ohms
Stub voltage	1.0V to 14.0V p-p, I-I, min. signal voltage (transformer coupled); 1.4V to 20.0V, p-p, I-I, min. signal voltage (direct coupled)

1.7.4. Architecture.

A terminal within a 1553 environment typically consists of four general functions. The four functions are: analog receiver/transmitter, digital bit and word processor, digital message processor, and the subsystem interface. Figure 1.7.4-1 illustrates the interface between these four functions. The unique addressing scheme used within the 1553 environment allows for up to 30 terminals to be connected within the network. 1773 network architecture is similar to that of a 1553 network, except for the use of fiber optics.

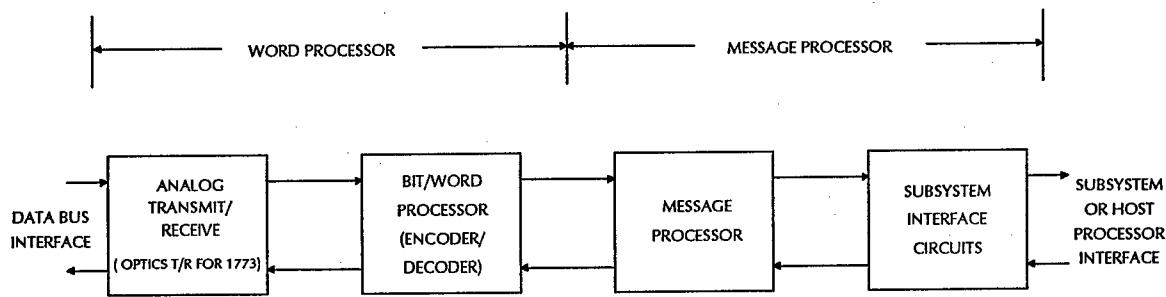


Figure 1.7.4-1. Terminal Functional Elements

Bus traffic within a 1553 network travels in one direction at a time (half-duplex) and is controlled by a terminal on the bus. Three types of terminals currently exist within a network. The following identifies and describes the primary function of these terminals and other elements within the network:

Bus Controller. The Bus Controller's main function is to provide data flow control for all transmissions on the bus.

Bus Monitor. The Bus Monitor's main functions are to listen to all addresses or a subset of addresses and to store the selected data for later use, and to observe transmissions on the data bus and to act on the data to solve problems that occur in the bus controller (generally used as a back-up bus controller).

Remote Terminal. The remote terminal's main function is to interface to the bus and respond to commands issued by the bus controller.

Data Bus Coupler unit. The Data Bus Coupler unit isolates the main bus from the terminal and thus prevents shorts.

Data Bus. The Data Bus is a twisted pair shielded cable for 1553 networks and fiber optics for 1773 networks.

1553 networks are made up of sensors or physical connections attached to the data bus. In order for all elements on the bus to communicate with each other, a bus topology must exist which allows all elements to have access to all transmissions. Bus topologies are illustrated in Figure 1.7.4-2. 1773 data bus topology is similar to that of a 1553 network.

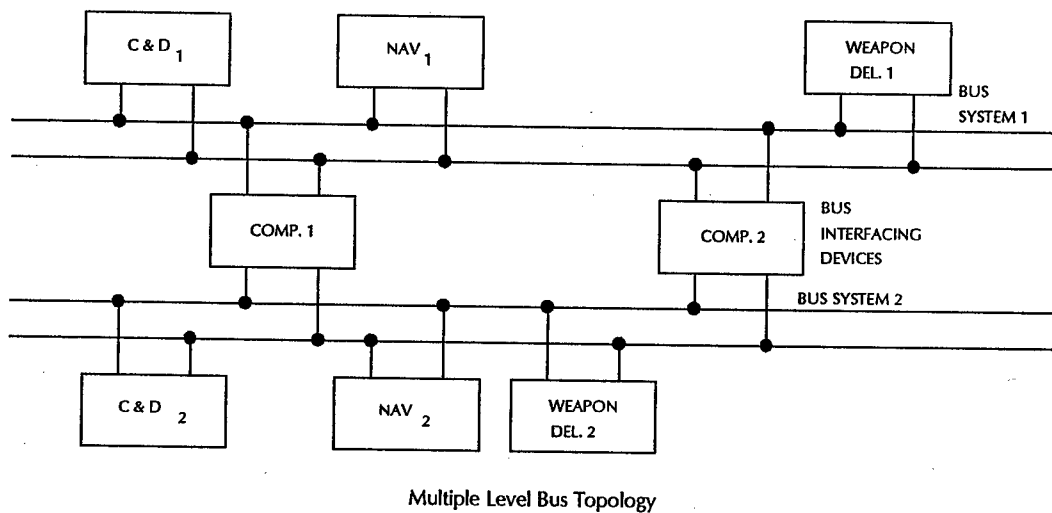
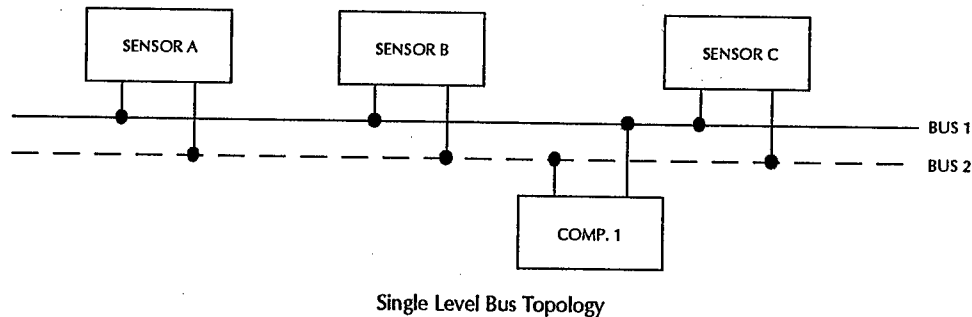


Figure 1.7.4-2. Bus Level Topology

Data Bus Topology. Data bus topology is the mapping of physical connections of each unit to the data bus. Two types of data bus topologies exist; single level and multiple level. Single level topology consists of all terminals connected to a single bus regardless of data bus redundancy. Multiple level topology (or hierarchical) consists of single level buses being capable of passing data on one bus system to another bus system.

Data Bus Control. Two schemes exist for controlling the data bus; stationary master and non-stationary master. In the stationary master scheme one master will control the data bus at all times. However, in systems with redundancy, there is usually a bus controller designated as backup to the primary bus controller. In a non-stationary bus master scheme, several bus controllers exist. Each bus controller is designated a specific time for controlling the bus. A bus control transfer mechanism is established to ensure each potential bus master gets its share of controlling the bus. Bus controller failure schemes must be established to ensure control of the bus is transferred to a healthy bus controller whenever the primary bus controller fails.

1.7.5. Protocol.

Basic message protocol can be divided into two groups; normal message transfers and broadcast message transfers. Message protocol is currently the same for 1553 and 1773 networks.

Normal Message transfers. Normal message transfer protocol requires that all error free messages received by a remote terminal be followed by the transmission of a remote terminal status word. This allows for validation of error free transmission.

Broadcast message transfers. Because broadcast message transfers are received by more than one remote terminal a different response scheme is required. A specific remote terminal can be designated as the sole responding terminal from a broadcast message, or the bus controller can poll each remote terminal to obtain its status.

All messages are initiated by the bus controller using command word(s). Messages are issued to a remote terminal through a message stream consisting of the remote terminal's address, direction of message transmission (transmit/receive bit), subaddress (destination within the specific remote terminal), and the word count. The command word is immediately followed by the appropriate number of data words identified in the command word. The receiving terminal validates the received message by transmission of a status word.

There are three types of words allowed in 1553 and 1773 environments; command word, status word, and data word. Each word consists of 16 bits plus sync pattern, plus parity bit (3 bit times long) and a one bit parity providing for a 20 bit long format. The command word sync and the status word sync are identical. The data word sync is the inverse of the command and status word sync. The bus controller does not have an address and parity is odd. Figure 1.7.5 illustrates the format and bit times for each word.

Command word. The command word uses bit times 1-3 for the sync, bit times 4-8 for the remote terminal address, bit time 9 for the transmit/receive direction indicator, bit times 10-14 for the subaddress/mode, bit times 15-19 for data word/count/mode code, and bit time 20 for parity. Command words can only be transmitted by a bus controller. The subaddress can act as a memory address pointer which contains the information requested for transfer to the bus controller.

Data word. The data word uses bit times 1-3 for the sync, bit times 4-19 for data, and bit time 20 for parity.

Status word. The status word uses bit times 1-3 for the sync, bit times 4-8 for the remote terminal address, bit time 9 for message error indicator, bit time 10 for the instrumentation indicator, bit time 11 for the service request indicator, bit times 12-14 are reserved, bit time 15 for the broadcast command received indicator, bit time 16 for the busy indicator, bit time 17 for the subsystem flag, bit time 18 for the dynamic bus control acceptance indicator, bit time 19 for the terminal flag indicator, and bit time 20 for parity.

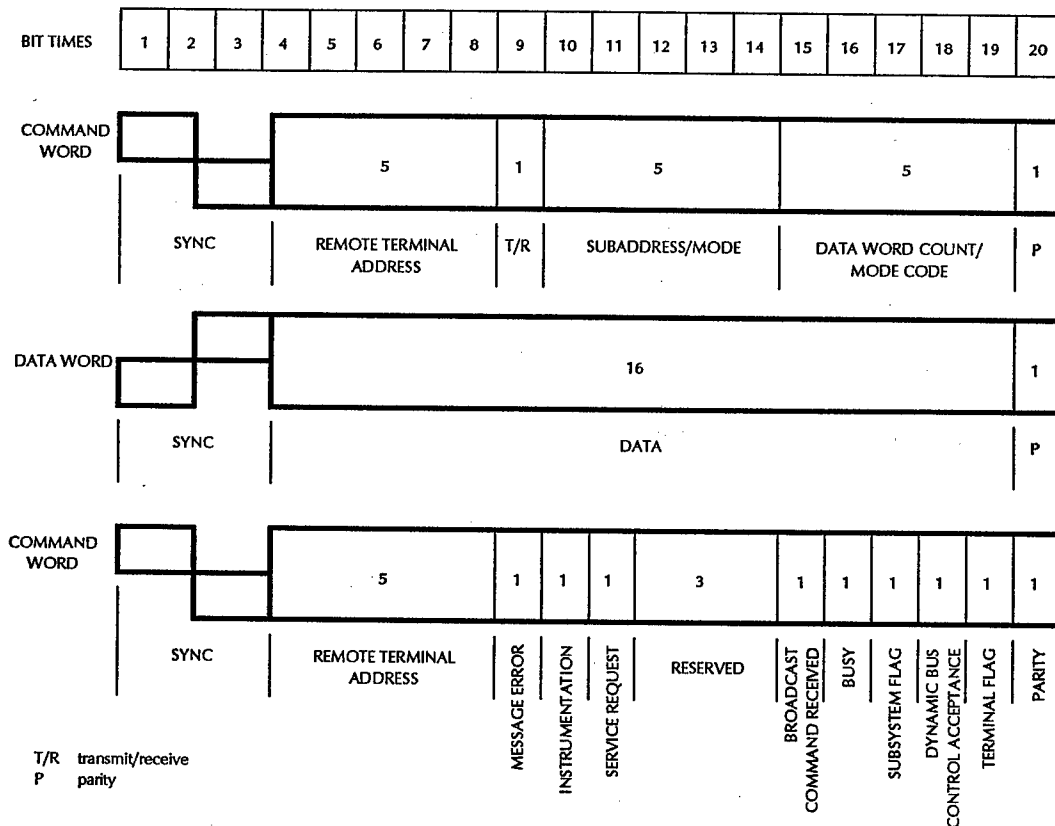


Figure 1.7.5. 1553/1773 Word Formats

1.7.6. Functions supported.

The four typical functions performed to support 1553 and 1773 environments are: analog receiver/transmitter (or Optics receiver/transmitter), digital bit and word processor, digital message processor, and the subsystem interface.

Analog (/Optics) Receiver/Transmitter. The Analog Receiver/Transmitter is the analog front-end required to interface the terminal's digital logic with the data bus. In 1773 networks, this functional element would be replaced by an optics receiver/transmitter. The analog receiver/transmitter contains the coupling transformer and fault isolation resistors for connection to the data bus. The receiver provides low noise rejection and a digital output compatible with the digital logic that follows in the bit and word processor. The transmitter drives the bi-phase modulated signal to form data word formats.

Encoder/Decoder. The encoder/Decoder analyzes the data bits and words required for data transfer on the receiver side of the terminal. The analog signal from the receiver is input to the decoder for proper manchester coding verification, error flagging, and parity checking. The encoder or data transmitter, generates the proper sync, data, and parity control for supplying the control signals for command, status, and data words to the analog transmitter.

Message Processor and Subsystem Interface. The message processor receives the data words, flagged if errors exist, from the encoder/decoder and analyzes the command word, address, data flow direction, and message length. The data is then prepared for/accepted from the subsystem interface.

Table 1.7.6 identifies the processes performed by each of the typical functional elements within a 1553 environment. A 1773 environment would provide similar functional elements.

1.7.7. Speed.

The speed at which the 1553 bus operates is based on the load being placed on the bus. The maximum number of bits per seconds is 1,000,000. Bus speed is currently 1 MHz. Current bus speed for 1773 is identical to the 1553 bus; 1 MHz.

Bus loading calculations consist of analysis of the following data:

- Message/type
- Words/message
- Overhead associated with each message type
- Overhead associated with mode codes
- Intermessage gap
- Average response time
- Overhead associated with non-stationary master bus controller passing

The overhead constants to consider are:

- | | | |
|----------------------|--|----------------|
| • command word | 20 microseconds (ms) | |
| • status word | 20 ms | |
| • response time | 2-10 (average 8) ms for 1553; 4-12 us for 1773 | |
| • intermessage gap | 2-30 (average 20) ms for 1553; 4.0 us for 1773 | |
| • mode codes wo data | 20 ms | (wo = without) |
| • mode codes w data | 40 ms | (w = with) |
| • data words | 20 ms | |
| • non-stationary BM | (48 minimum) ms | |

Table 1.7.6. Terminal Functional Processes

Analog Receive/Transmit	BIT/Word Processor (encoder/decoder)	Message Processor	Subsystem Interface Circuits
Receive	Receive	Receive	Channel selection (notes: 1,2)
Signal limiting	Receiver	Command word decode (note: 1)	Data sampling (notes: 1,2)
Filter	Sync detection	Status word decode (note: 3)	Conversion (notes: 1,2)
Threshold detection	Data detection	Address recognition (note: 1)	Subsystem timing (notes: 1,2)
Transmit (note: 2)	Manchester error detection	Mode execution (note: 1)	Buffer registers (notes: 1,2)
Driver (note: 2)	Parity check	Word count recognition	Calibration (notes: 1,2)
Transmitter control (note: 2)	Bits/word count	Message error detection	Self-test (notes: 1,2)
Timer (note: 2)	Transmit (note: 2)	Transmit (note: 2)	DMA to host memory
Common	Transmit control (note: 2)	Word count (BC) (notes: 2,3)	Interrupt lines
Bus coupling	Sync/data encode (note: 2)	Status register (note: 2)	Control lines
	Parity generation (note: 2)	Built-in test	
	Clock generation	Subsystem interface control	
		Data address	
		Control registers	
		Memory buffer	

Where:

- Note1: not used in Bus Controller
 Note2: not used in Bus Monitor
 Note3: not used in Remote Terminal

1.8. High Speed Data Bus.

1.8.1. Overview and Intended use.

The HSDB provides a high speed message passing capability between architecture elements. The HSDB is a fiber optic, linear token passing data bus capable of a 50 million bits per second (MBPS) transfer rate. Typical application of the HSDB is communication of high speed data, control, and message communication within military and commercial applications.

The bus may be implemented as a single bus architecture, however, military applications generally implement a dual -redundant bus architecture to enhance system reliability and survivability.

1.8.2. Current Status.

Though the High Speed Data Bus (HSDB) is being used in commercial applications, interest is high for its use in advanced military platforms. The Joint Integrated Avionics Working Group (JIAWG) is developing a requirements specification for a linear token passing multiplex data bus and associated data bus protocol. Advanced avionic platforms are envisioning the use of the HSDB to reduce vehicle weight.

Testing Application.

Maintenance and diagnostic efforts for advanced avionic platforms are using the HSDB to communicate at high speed control information between system elements, and diagnostic status/test information between subsystems. Typical system use of the HSDB in military applications consists of a dual, redundant bus configuration to gather vehicle information from engine diagnostics, subsystem performance, radar data, and testing applications. The HSDB is also being used to provide a communications path for rapid transmission of system software in support of reconfiguration applications.

1.8.3. Interface / Number of Pins.

General interface requirements characteristics associated with using a HSDB within system designs fall into four specific groups:

- common characteristics,
- transmitter characteristics,
- receiver characteristics,
- and transmission media characteristics.

Common. Common characteristics associated with a transmitter, receiver, and medium include the station I/O ports, optical wavelength and spectral bandwidth, bus data rate, minimum preamble size, bus signaling rate, and minimum inter-transmission gap. Typical units associated with these elements are identified in Table 1.8.3.

Table 1.8.3. Bus Element Characteristics

Description	Units	Requirement
Encoding Method		Manchester II
Data Rate	MBPS	50 +/- .01%
Signaling Rate	MBaud	100 +/- .01%
Nominal Bit time	nS	20
Minimum Duration Between Transitions	nS	10
System Minimum Inter- transmission Gap	nS	280
Preamble Minimum Size	Bit times	16
Optical Wavelength Upper/Lower	nM	800-lower, 880-upper
Spectral Bandwidth	nM	60

Transmitter. Typical characteristics associated with a transmitter include signal power level (high/low), signal leakage power, maximum rise/fall times, pulse width distortion, nominal bit time, minimum signal duration, and data streaming timer.

Receiver. Typical characteristics associated with a receiver include signal power input, operating range, inter-transmission dynamic range, maximum rise/fall times, pulse width distortion, nominal bit time, maximum signaling rate, and maximum bit error rate.

Media. Typical media characteristics associated with a fiber optic bus include core diameter, maximum total dispersion, and min/max optical attenuation.

1.8.4. Architecture.

The bus consists of a set of stations connected by a broadcast transmission medium. Each station which transmits shall be heard by all active stations. Stations accept transmissions based upon either physical or logical addressing mechanisms. Physical network configurations can exist with bus lengths of up to 1000 meters.

Access to the station shall be controlled by a token. The token shall be continually passed around a logical ring superimposed on a linear bus. A station shall have sole access to transmit on the media when it has possession of the token.

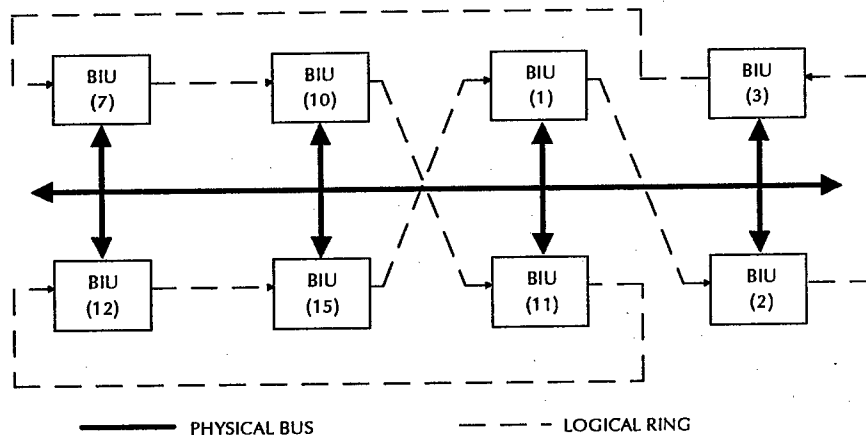


Figure 1.8.4. High Speed Data Bus Architecture

Station Registers. Each station shall support the following registers:

- Next station Address range 0 to 127
- Maximum Station Address range 0 to 127
- Physical Station Address range 0 to 127

Station States. A station shall support the following functional states:

State	Name	General Actions
0	Off-Line	no transfers or receptions
1	Idle	waiting for token, msg, or BAT
2	Claim token	try to claim the token
3	Check token address	reception of a valid token frame
4	Send messages	when the token is received
5	Pass token	pass the token to the next station
6	Check token pass	verify token is passed
7	Check message address	reception of a valid message frame
8	Receive messages	reception of valid data

1.8.5. Protocol.

Protocol within an HSDB environment can be thought of as consisting of the state of the transmission medium and the message to be transmitted or received.

HSDB medium states consist of the following:

- **Quiet/Idle** shall indicate the state of the medium when no bus activity is present.
- **Active** shall indicate the state of the medium when a signal is being transmitted. Valid activity is defined by the presence of at least three signal transitions in a period of 4 bit times.

Messages transmitted or received over the HSDB medium shall consist of the following:

- **Preamble** is sent at the beginning of every message and is used to stabilize the receiver to the incoming signal.
- **Message frames** contains a Start delimiter, frame control, source address, message destination address, word count, information, message frame check sequence, and an End delimiter. A Start delimiter defines the beginning of a frame. This delimiter allows the receiving station to synchronize itself to the frame. The End delimiter defines the end of a frame. This delimiter allows the receiving station to terminate the reception of the frame. Data within the message frames consists of logical '0' or '1' state encoded in Manchester II biphasic format.
- **Token frame** consists of a start delimiter, one bit frame control (set to zero), token destination address, token frame check sequence, and end delimiter fields.
- **Claim token frame** consists of a start delimiter, frame control, source address, fill words, and end delimiter fields.

Each frame shall consist of a sequence of fields. There shall be no gaps between the individual fields in a single frame.

1.8.6. Functions supported.

Stations supporting a HSDB environment shall support the following functions.

- **Message Priority** priority of messages shall consist of four levels; 0-highest, 1, 2, and 3-lowest
- **Source address** the address of the station transmitting a claim token or message frame; this is the physical station address (PSA).
- **Message destination address** uniquely identifies the message frame's destination address. It can specify a logical or physical address. There is an indicator as to the type of addressing used (physical or logical).
- **Broadcast** each station shall support a broadcast mode in which a message is received by all stations.
- **Word count field** specifies the total number of 16 bit words transmitted. Field size is 0 to 4096 words.

- **Message frame check sequence** the ability to verify proper transmission of each message shall be supported. This can be accomplished by having the transmitter insert a message frame check sequence into the message sent and having the receiver verify that the sequence is correct.
- **Token frame check sequence** is similar to the message frame check sequence except a token frame check sequence is sent and verified.

Within a HSDB environment, timers are used to limit the amount of time a station is in control of the bus.

- **Holding Timers** is the amount of time a station may transmit message frames after receiving the token is limited. This amount is user defined and can range from 0 to 65,535 microseconds. General default value is 50 microseconds.
- **Rotation Timers** is the amount of time a station may transmit frames for a particular priority level or any lower priority level. Priority levels are from 1 to 3. This amount is user defined and can range from 0 to 65,535 microseconds. General default value is 400 microseconds for priority 1, 300 microseconds for priority 2, and 200 microseconds for priority 3.
- **Ring Admittance Timers** controls when a station may be allowed to enter the logical ring. This amount is user defined and can range from 0 to 6,553.5 milliseconds. General default value is 100 milliseconds.
- **Token Pass Timer** is a timeout value set to determine whether or not the token was successfully passed on the logical ring. Valid range is 0 to 10.20 microseconds with a default of 10.20 microseconds.
- **Bus Activity Timer** is a timeout value set to allow a station to claim the token after a specified time of no bus activity. Valid ranges are 0 to 2047 microseconds.
- **Claim Token Limit Counter** identifies the maximum number of times the station is allowed to send a claim token frame without transitioning from the claim token state. This amount is user defined and can range from 0 to 127 with a default of 4.

1.8.7. Speed.

The data rate of the HSDB is typically 50 +/- 0.01% million bits per second (MBPS). The signal baud rate for continuous transmission on the fiber optic medium shall be 100 +/- 0.01% MBaud. The normal duration between transitions shall be 10 +/- 1 nanoseconds. The optical receiver shall operate with a minimum inter-transmission gap of 280 nanoseconds. The maximum inter-transmissions gap shall be 500 nanoseconds.

1.9. IEEE P1394.

1.9.1. Overview and Intended use.

IEEE P1394 is a high speed serial bus designed for low cost systems. It is expected to be used for interconnecting remote peripherals to high performance backplanes (such as NuBus, Futurebus+, and SCI). It is a powerful, low cost, high-speed, 2-wire serial bus which is compatible with parallel buses. The IEEE P1394 bus provides support for multi-bus configuration, dynamic addressing and experiences low communications overhead. Figure 1.9.1 illustrates an IEEE P1394 serial bus physical topology. Bus topology is divided into a backplane environment and a cable environment. There is no requirement that the bus have a set of environments. All bus nodes may be connected directly to the backplane, or cable, or any combination of the two.

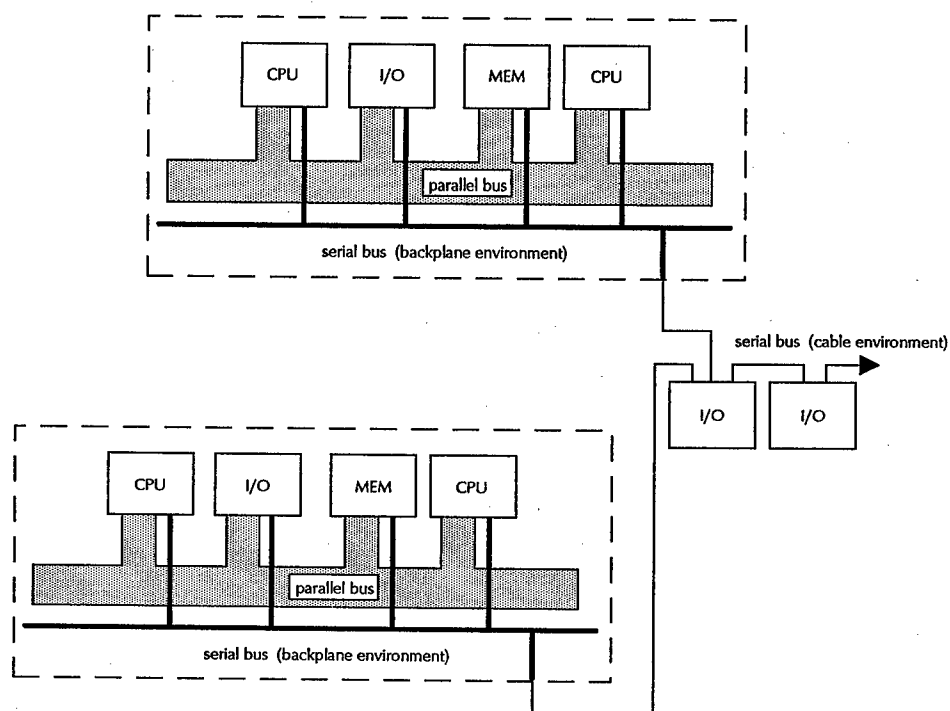


Figure 1.9.1. P1394 Serial Bus Physical Topology

1.9.2. Current Status.

The Microprocessor and Microcomputer Standards Subcommittee working group, sponsored by the IEEE Computer Society, is currently working on the High Speed Serial Bus P1394 standard. Work is currently being performed to interface the proposed standard with the IEEE 1149.1 standard.

Testing Application.

The IEEE P1394 high speed serial bus is envisioned to provide a possible low cost communications interface to the IEEE 1149.1 Test Access Port and Boundary Scan Architecture standard. Via the IEEE P1394 bus, test control and data could be transmitted to the IEEE 1149.1 testing environment to verify/test system, module, and

integrated circuit designs. Results from testing would then be returned to higher applications for test status assessment.

1.9.3. Interface / Number of Pins.

The IEEE P1394 bus interface consists of 3-pair shielded cables with terminators, transceivers, and logic dedicated to each port. Two of the three twisted pairs are individually shielded and carry the data signals. The third pair carries the power. The twist rate of the cable is 36 to 44 per meter, with the maximum cable length being 2.0-m. Currently, there is no minimum cable length.

The backplane interface includes the driver and receiver as well as the particular signal lines. Devices connected to the bus backplane utilize two-electrical signals identified by the various ANSI/IEEE bus standards. Drivers and receivers for these signals observe conventions established by parallel bus standards; e.g., Futurebus, Fastbus and SCI.

1.9.4. Architecture.

The architecture of the IEEE P1394 serial bus consists of entities called nodes; a node being an addressable entity which can be independently reset and identified. Figure 1.9.4 illustrates a module design which utilizes nodes connected to the serial bus. Several nodes may be co-resident on a single module, and more than one function may be co-located on a single node. Nodes are tied together with cables that lace a single connector plug on each end. Two nodes connected together (full-duplex path) with a cable is called a physical connection, and there can be up to 6 physical connections separating any two nodes. The total cable length between any two nodes must be less than 10 meters.

The address scheme observed by the serial bus uses 64-bit fixed addressing. The upper 16 bits of each address is the node_id (allowing 64k nodes in a system). The higher-order 10 bits specify the bus_id and the lower 6 bits specify a node_offset. This allows an address scheme for 1022 buses each with 63 nodes to be accessible.

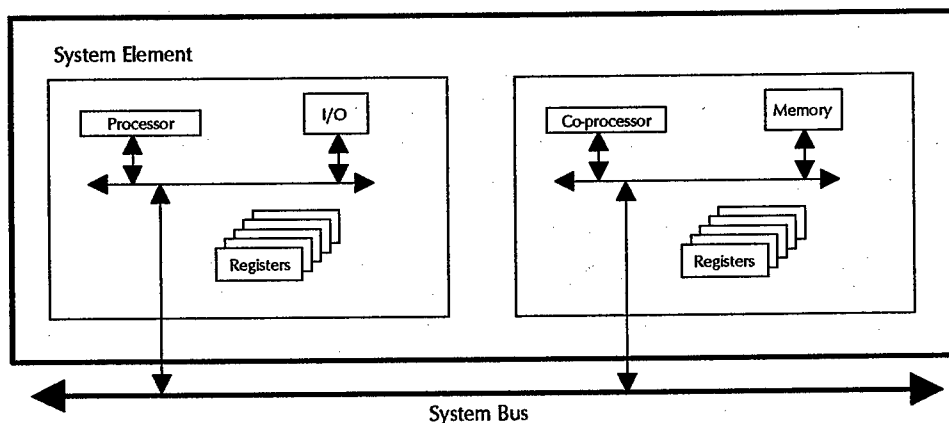


Figure 1.9.4. Module Node Architecture

1.9.5. Protocol.

The serial bus protocols are described as a set of three stacked layers; transaction layer, link layer, and physical layer. A fourth element, the serial bus management, provides other applications required by bus nodes. Figure 1.9.5 illustrates the connectivity between these four elements.

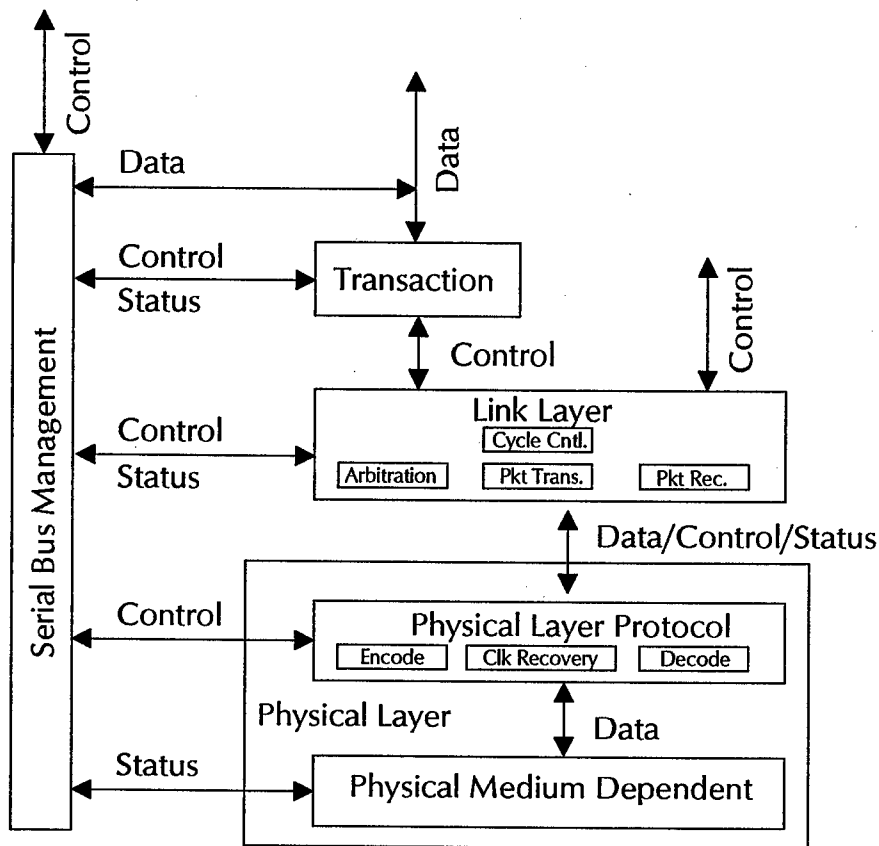


Figure 1.9.5. IEEE P1394 Bus Protocol Element Connectivity

Transaction Layer. The Transaction Layer defines a complete request-response protocol to perform the bus transactions. Bus transaction types and actions are listed in Table 1.9.5. Data transfers can be from 1 to 2048 bytes with no address restrictions.

Table 1.9.5. Transaction Layer Transaction Types/Actions

Transaction Types	Description
Read	data is transferred from a responder back to a requester
Write	data is transferred from a requester to one or more responders
Lock	data is transferred from a requester to a responder, operated on by the responder, and then transferred back to the requester
Transaction Actions	Description
Request	the action taken by a requester to start the transaction
Indication	the reception of a request by a responder
Response	the action taken by the responder to finish the transaction
Confirmation	the reception of the response by the requester

Link Layer. The Link Layer defines a one-way data transfer service to the Transaction Layer. It provides access to the medium, addressing, data checking, and data framing. One Link Layer transfer is called a 'subaction'. There are two types of subactions: Asynchronous and Isochronous (or channel).

- **Asynchronous** 0 to 2048 bytes of data and several bytes of Transaction Layer information are transferred to an explicit address. The subaction has three parts; 1) arbitration sequence, 2) packet transmission, and 3) acknowledgment
- **Isochronous Subaction** 0 to 256 bytes are transferred without explicit addressing. Channel transfers are used for cycle responses where a form of time division multiplexing provides an implicit addressing.

The Link Layer operations also have the request, indication, response, and confirmation actions:

- **Request** the action taken by a link requester to transmit a packet to a link responder
- **Indication** the reception of a packet by a link responder

- **Response** the transmission of an acknowledgment by a link responder
- **Confirmation** the reception of the acknowledgment by the link requester

Physical Layer. The Physical Layer translates the logical symbols used by the Link Layer into actual physical signals on the different serial bus media. The Physical Layer has two sublayers: the Physical Protocol Sublayer, and the Physical Medium Dependent Sublayer. The backplane and cable environments have different Physical Medium Dependent sublayers. The Physical Protocol sublayer performs the data encoding, decoding, and clock recovery operations. The Physical Medium Dependent sublayer specifies the physical (connectors and cables) and electrical interface (transceivers).

Serial Bus Management. Serial Bus Management provides the basic control functions and standard control and status registers needed by nodes on the bus. Bus Management provides many management facilities:

- **Arbitration number/address assignment**
- **Cycle master arbitration** the candidate with the most accurate clock will become master
- **Isochronous channel assignment**
- **Error control** the bus will detect that there is an error but will not necessarily retry any transfer.

1.9.6. Functions supported.

In addition to bus arbitration, the following services are provided by the four elements associated with serial bus protocol.

Physical Sublayer Services.

Service	Description
control request	to perform low-level actions or obtain status
control confirm	completion
arbitration request	to send an arbitration symbol
arbitration confirm	return status of the bus
state indication	inform Link Layer of bus state change
data request	inform Link Layer to send a data symbol
data indication	inform Link Layer of arrival of data symbol

Link Layer (LL) Services.

Service	Description
control request	to perform low-level actions or obtain status
control confirm	completion
status indication	inform bus management of status change or event
sync request	
sync indication	indicate cycle sync packet is received
ISO request	from upper layer to request LL to send ISO data packet
ISO indication	inform upper layer of arrival of ISO packet
data request	from Transaction Layer to request LL to send async data packet
data confirm	completion
data indication	defines transmission of data to the local Transaction Layer
data response	returned status (complete, pending, busyA, busyB)

Transaction Layer (TL) Services.

Service	Description
control request	to perform low-level actions or obtain status
control confirm	completion
status indication	inform bus management of status change or event
data request	from application to request TL to send a request
data confirm	completion
data indication	informs the application of the arrival of a transaction request
data response	returned a response

Serial Bus Management Services.

Service	Description
control request	from application to request Bus Manager to perform various control actions
control confirm	completion
control indication	indicates the status of the bus has changed

1.9.7. Speed.

The IEEE P1394 Serial Bus supports variable speed data transmissions of approximately 40 Mbit/sec between nodes separated by distances up to 10 meters.

2. TEST BUS EXTENSIONS

2.1. IC.

The use of the IEEE 1149.1 standard in integrated circuit (IC) designs and integration is becoming more prevalent throughout industry. Extensions to the standard are being investigated and developed to support extended testing capabilities. The envisioned benefits from these efforts are:

- the ability to establish a consistent test instruction set throughout company devices,
- compatible testing modes,
- reduced complexity in development of test and maintenance software tools,
- and reduced efforts in system integration and debug.

The following subparagraphs describe some of the work being performed to extend the capabilities of the IEEE 1149.1 in support of testing applications at the integrated circuit level.

2.1.1. Internal Scan.

One of the most widely used design-for-test practices for IC design is internal scan design. Using internal scan design, an IC's application logic can be easily tested by scanning test patterns through the design via a serial scan port to verify its logic implementation.

The IEEE 1149.1 standard was designed primarily to provide a boundary scan test architecture in ICs to facilitate board level testability. However, the 1149.1 architecture was designed to be expandable so that other IC level test structures could be supported as well. One of the test structure extensions that can be included in the 1149.1 architecture is internal scan design. This section illustrates how internal scan design can be included in the 1149.1 architecture and accessed to effectuate testing of an IC's application logic.

BASICS OF INTERNAL SCAN DESIGN.

In Figure 2.1.1-1 an example IC logic design is shown consisting of modules A, B, C, D, E and F. All modules are sequential and operate synchronous to a common clock input (CLK). Module A receives the primary inputs (PI) to the IC and module F supplies the primary outputs (PO) from the IC. Intermediate modules B, C, D, and E receive inputs from each other and from module A, and output to each other and module F. Module A is directly controllable via PI and module F is directly observable via PO. However, the inputs to and output from the intermediate modules are not directly controllable or observable via PI or PO.

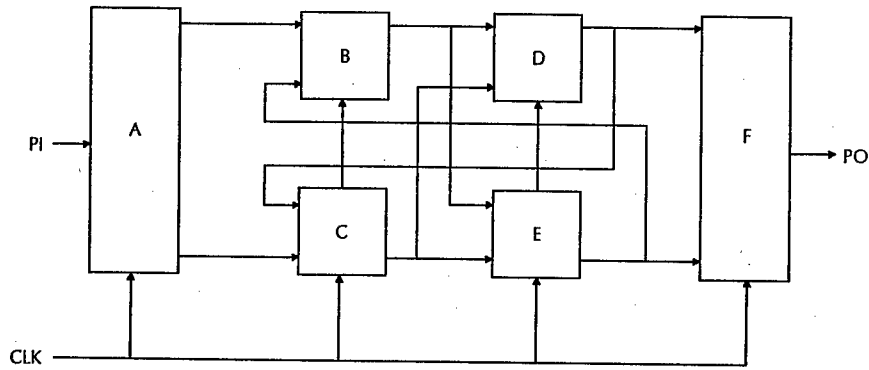


Figure 2.1.1-1. Non-Scannable IC Logic Design

In Figure 2.1.1-2 an example circuit model of each module is shown. Each module consists of input combinational logic and output flip flops (FFs). The state of the output FFs (OUT1-OUTx) is determined by the output response of the combinational logic to input stimulus (IN1-INx) at the clock time. What makes testing of the intermediate modules difficult from the PI inputs and PO outputs is the fact that the state of each intermediate module is dependent upon both external input from PI and from the internal state of each module. Due to this situation, perhaps tens of thousands of test patterns must be input to PI and output from PO to fully test the intermediate modules.

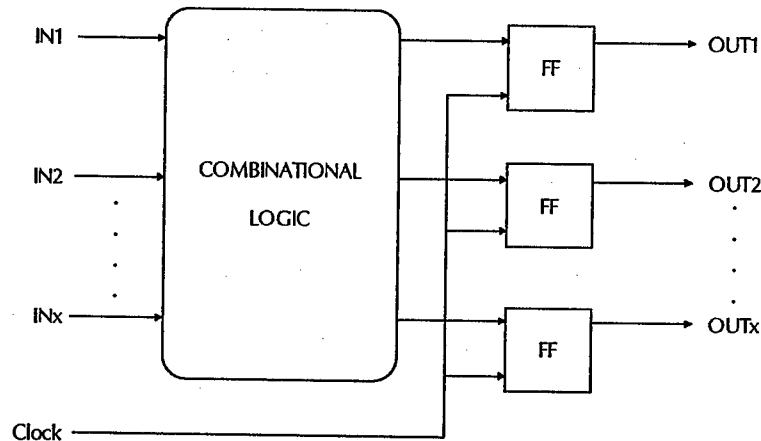


Figure 2.1.1-2. Non-Scannable Module Circuit

To overcome this testing roadblock, modules B, C, D, E, and F can be designed to be scannable as shown in Figure 2.1.1-3, and module A can be designed to be scannable as shown in Figure 2.1.1-4. Module A, of Figure 2.1.1-4, includes scan FFs and multiplexers on its inputs to allow the module to be controlled, during scan testing, independent from the PI inputs. The conversion from the non-scannable module of Figure 2.1.1-2 to the scannable modules of Figure 2.1.1-3 and 2.1.1-4 is simply the replacement of the non-scannable FFs with scannable FFs, and the addition of input scan FFs and multiplexers to module A.

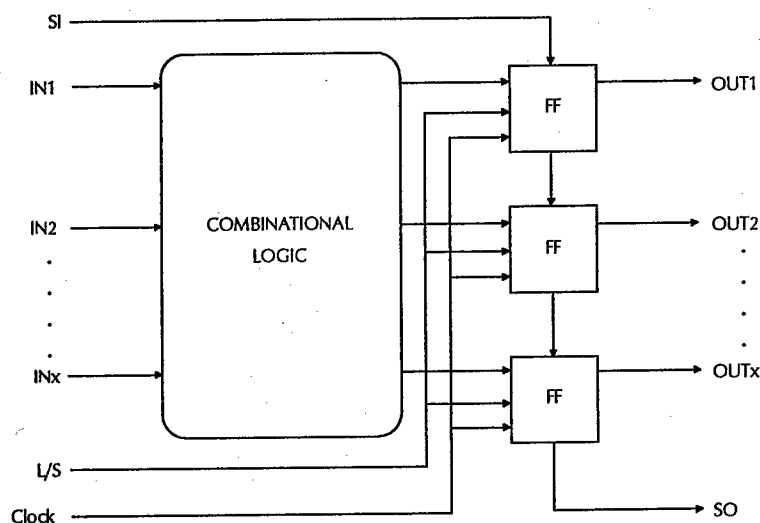


Figure 2.1.1-3. Scannable Modules B,C,D,E, and F

In Figure 2.1.1-3 and 2.1.1-4, a load/scan control signal (L/S) is required to regulate the behavior of the scan FFs. When the L/S signal is in load mode the scan FFs operate as normal FFs, but when it is in scan mode the scan FFs shift data from their serial input (SI) to serial output (SO) in response to the clock input. Also, module A of Figure 2.1.1-4 requires a run/test control signal (R/T) to regulate the input multiplexers to output either PI data or scan FF data to the combinational logic.

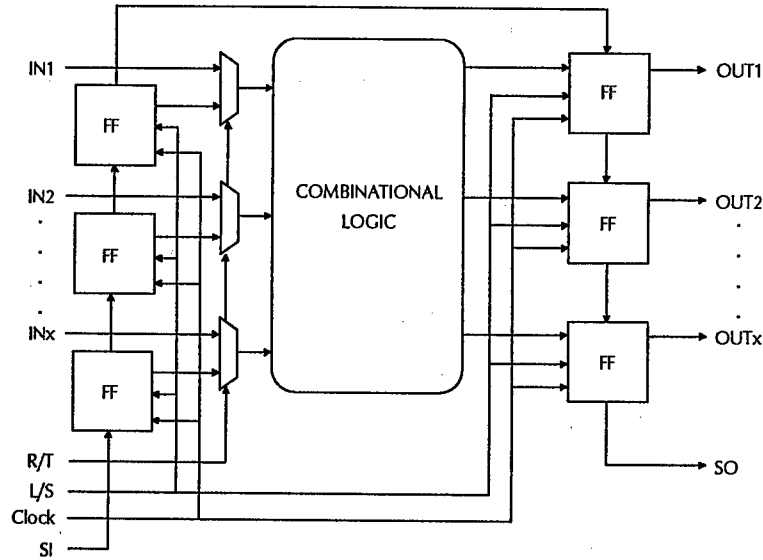


Figure 2.1.1-4. Scannable Module A

In Figure 2.1.1-5, a view of the IC's logic design is shown using the scannable modules of Figure 2.1.1-3 and 2.1.1-4. The modules are connected serially together to form a scan path from the IC's serial input (SI) to serial output (SO). The changes required to the IC's architecture, in addition to the scan FFs, include; a clock multiplexer, a run/test input (R/T), a test clock input (TCK), a load/scan input (L/S), a scan in input (SI), and a scan out output (SO). R/T regulates the input multiplexers of module A and also controls whether the system clock (CLK) or the test clock (TCK) is output from the clock multiplexer to the module scan FFs. During normal IC operation, R/T is set to run mode and CLK times the module scan FFs. During IC test operation, R/T is set to test mode and TCK times the module scan FFs. During test mode, the L/S input regulates the operation of the module scan FFs, allowing them to load and shift out test data. During scan operations, SI inputs serial data to the ICs scan path and SO outputs serial data from the ICs scan path.

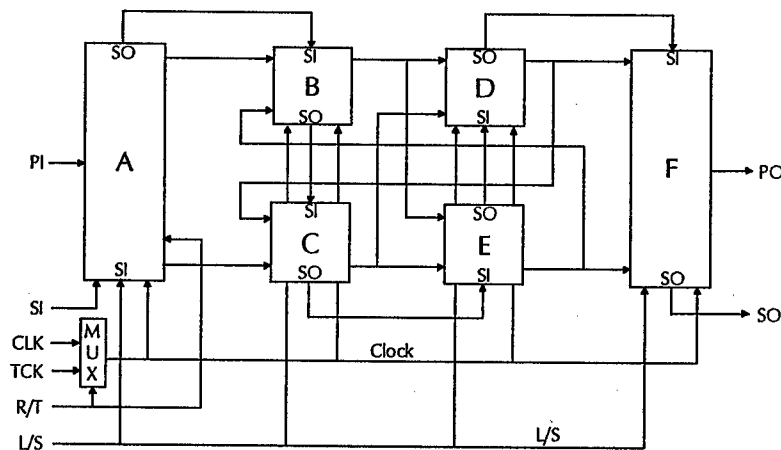


Figure 2.1.1-5. Scannable IC Logic Design

When testing is not required, the R/T and L/S signals are set to allow the IC to operate normally. When testing is required, R/T connects the TCK to the modules and the L/S signal is operated, in coordination with the TCK, to enable the scan path to load data from its functional input, then shift data from SI to SO. The effect of the test mode is that it simplifies the IC design to where all combinational logic sections are directly controllable and observable via the scan path FFs. By operating the scan path to input stimulus to the combinational logic and to capture response from the combinational logic, the IC design can be easily and thoroughly tested. Also the scan path allows testing the IC using a minimum number of test patterns, far fewer than the number of test patterns required to test the IC via PI and PO. Further, the test patterns applied via the scan path can be generated by automatic test pattern generation tools, eliminating the time consuming task of manually creating the test patterns.

ACCESSING INTERNAL SCAN VIA IEEE 1149.1

Internal scan design can be accessed via the 1149.1 architecture by designing the IC's application logic to be scannable and defining an 1149.1 instruction for internal scan access. In Figure 2.1.1-6 an IC is shown including the 1149.1 architecture. The required components of the 1149.1 architecture include the boundary scan register, bypass register, instruction register, and test access port (TAP). The serial interface to the 1149.1 architecture includes a test clock (TCK), a test mode select (TMS), a test data input (TDI), and a test data output (TDO).

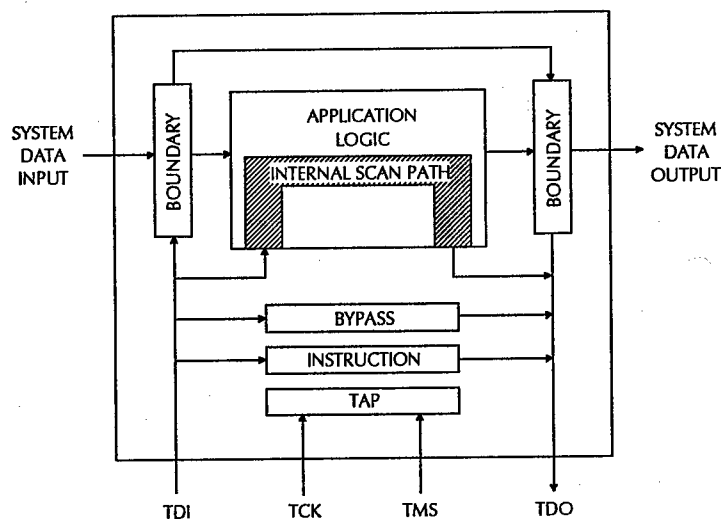


Figure 2.1.1-6. Internal Scan in 1149.1 Architecture

In addition to the required 1149.1 components, the IC's application logic includes an internal scan path, as described previously, that can be accessed to test the IC design. The internal scan path is accessed by loading a user-defined instruction into the instruction register that connects the internal scan path between TDI and TDO. The instruction also places the IC in a test mode and enables the scan path to be controlled

by the TAP, similar to way the previous scan path was controlled by the R/T, L/S and TCK signals.

During internal scan testing, the TAP receives control input from TCK and TMS to cause test patterns to be scanned into the ICs internal scan path from TDI to TDO to test the application logic. While scan testing of the application logic is in progress, the boundary scan register acts to isolate the application logic from external interference, and to prevent signals generated by the application logic during the test from being output to external neighboring ICs.

Internal scan design is an effective method of testing an IC's application logic. The 1149.1 standard, while primarily designed for structural testing of board assemblies, easily accommodates internal scan test approaches. This section has illustrated the concept of internal scan design and how access to internal scan design can be achieved via the 1149.1 standard.

2.1.2. BIST.

Built-In-Self-Test (BIST) is a design for testability (DFT) technique that has gained significant acceptance within the electronics industry as a means to deal with the problems of testing complex chips, boards, multichip modules, and systems. BIST approaches and architectures can complement boundary scan architectures to provide a complete PWB and system test environment. The implementation of both techniques is becoming more widespread and BIST extensions to boundary scan implementations are emerging.

Traditionally, when discussing BIST, chip-level BIST based on a linear feedback shift register (LFSR) architecture typically comes to mind first. However, as more complex electronics architectures have emerged, module and system level BIST architectures built on top of boundary scan architectures are taking their place along with device internal BIST. The IEEE 1149.1 test bus can be used to control device internal BIST, BIST on a module logic cluster, or BIST on a module memory array. In addition to conventional LFSR based approaches, a module may embed an 1149.1 scan controller to allow embedded application of deterministic patterns.

BIST Techniques.

There are many techniques used to implement BIST in components. A few of the most common are pseudo random pattern generation (PRPG), parallel signature analysis (PSA), serial signature analysis (SSA), and self-test state machines. All these techniques or combinations may be used for BIST.

Test pattern stimulus is typically applied using PRPG, which is implemented by a free-running linear feedback shift register (LFSR) that generates a unique pseudo random sequence of test patterns. This pseudo random sequence of test patterns can be applied to the circuit under test at-speed to exercise the logic with minimal overhead.

PSA (or SSA) usually is used in conjunction with PRPG to capture and compress the output states of the circuit under test. PSA or SSA also is constructed from an LFSR, which compresses a fixed number of parallel or serial output patterns into one unique

signature. At the completion of the test sequence, the signature is compared to a known good signature to determine if the test passed or failed.

State machines also are used commonly to test circuitry that requires specific test sequences or patterns. This usually includes random access memory (RAM) or combinational logic. One example is the case of a RAM or register array that requires the address to be sequenced as test patterns are written to and read from the array. State machines also are used to start and stop PRPG/PSA based tests.

BIST Using the 1149.1 Test Bus and Boundary Scan.

When discussing BIST as related to 1149.1 test bus architectures, two areas need to be addressed. The first area relates to embedded architectures which can autonomously control embedded IEEE 1149.1 tests and the IEEE 1149.1 test logic. The second area focuses on architectures which complement basic boundary scan logic for algorithmic pattern generation and compression (e.g. PRPG, PSA).

In the simplest case, an IC, module, or system may be tested under external control using an 1149.1 test bus controller to control the initialization, application, and comparison of test sequences. This may be accomplished under discrete control by a 4-bit port of a microcontroller or more easily by a Test Bus Controller (TBC) application specific integrated circuit (ASIC). With the latter approach, the bus states can be controlled in hardware and would not have to be sequenced explicitly. Additionally, the TBC device could incorporate 'intelligence' to increase data throughput and further off load the microcontroller or processor.

An integrated solution is possible by embedding the TBC in the system or module design. This method, as shown in Figure 2.1.2-1, provides a capability for autonomous module or system BIST. There are many methods to implement an 1149.1 embedded test controller. Two methods include a simple state machine test controller and a programmable test controller. Both can provide an autonomous embedded 1149.1 based test for system startup self-test or commanded self-test. An embedded controller must sequence the UUT through TAP states, apply stimulus data to TDI, capture response data from TDO, mask any "don't care" bits, and compare the masked response data to the expected data.

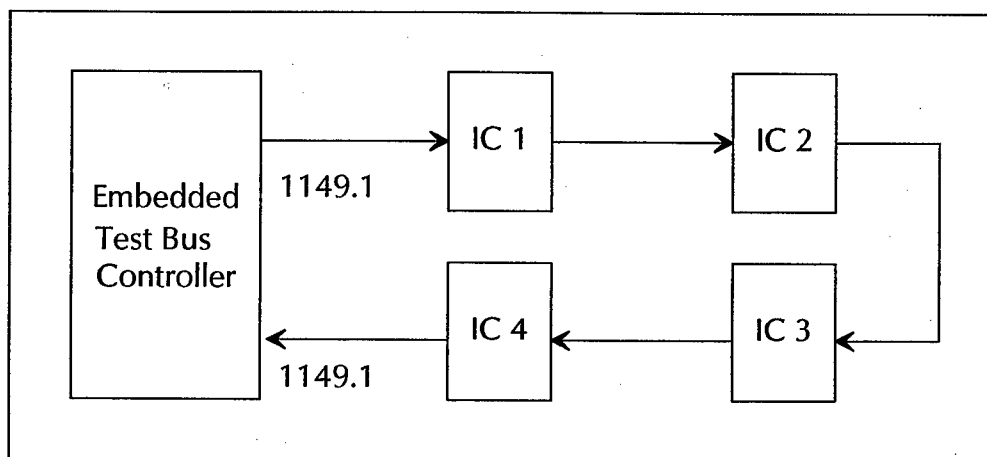


Figure 2.1.2-1. Embedded 1149.1 Test Controller

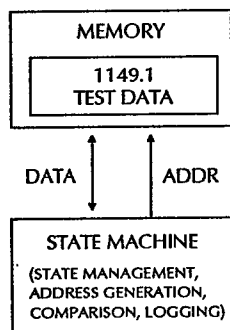
A state machine based 1149.1 controller is simple if only a static configuration is needed. This type of controller can be designed in an ASIC or FPGA. An external ROM could be used to store the test data. The state machine generates addresses for the test data in ROM and control comparator logic compares expected data to actual data. The test data stored in ROM includes the TMS, TDI, TDO, and MASK values.

Alternately, a programmable test controller (PTC) can be implemented which is flexible and intelligent. A programmable test controller can utilize the intelligence and programmability of a microprocessor to retrieve test stimulus from memory, write test stimulus and control commands to the test bus controller, and read/compare/log response data.

A PTC can provide several advantages over a simple state machine driven test controller. Because of the intelligence and flexibility of a PTC, tests can be executed with different logging modes, execution modes, and data compression.

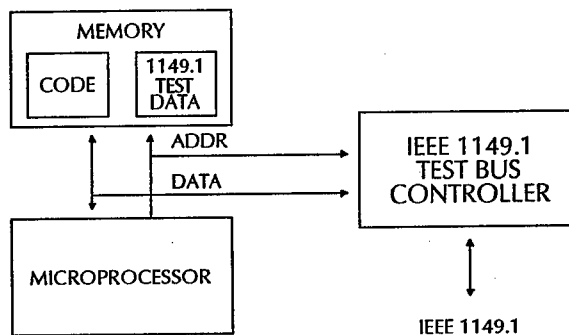
A PTC can be implemented with a microprocessor or microcontroller, a TBC device, and memory to store code and data. The test bus controller device performs the parallel to 1149.1 serial translations for both transmitted and received data and manages the 1149.1 TAP states of the UUT. The processor executes control code to fetch test data from memory, transfer the data to the TBC, read received test data from the TBC, and perform data comparison and/or logging. Basic block diagrams of a state machine based 1149.1 controller and a programmable test controller are shown in Figure 2.1.2-2.

STATE MACHINE BASED TEST CONTROLLER



IEEE 1149.1

PROGRAMMABLE TEST CONTROLLER



IEEE 1149.1

Figure 2.1.2-2. Embedded 1149.1 Test Controller Architecture

As mentioned above, a fixed length pattern file may be stored in onboard memory and applied by the TBC. This method requires virtually no processing and, therefore, allows a simplified TBC controller to be used. The drawback in this case is the amount of memory required to store a potentially large test pattern set. For the second option, a processor could apply algorithmically derived patterns and collect and/or compress the results. In this case, very little memory is required to store test patterns, but a 'smarter' controller is required to handle pattern application and collection. The pattern application rate of the latter option could be slower if extensive processing is necessary.

When dealing with scan based tests, the data sets can quickly become extremely large if the scan data for all devices in the target device's scan ring must be stored. Storing this amount of data may become unfeasible for an embedded application. Fortunately, a smart PTC can be implemented which scans the complete ring, but the test data need only contain stimulus and expected response data for the device or devices under test. This can be realized if the PTC manages the placement of stimulus and response data. The PTC inserts leading and/or trailing bits to the stimulus and response data on-the-fly and therefore the test data need not contain unused or don't care data. The actual data scanned is the combination of any leading bits, the test data, and any trailing bits. The result is a test data file which is more compact because the leading/trailing bit offsets are generated on-the-fly rather than stored in the test data file. An example of on-the-fly leading and trailing bits is shown in Figure 2.1.2-3.

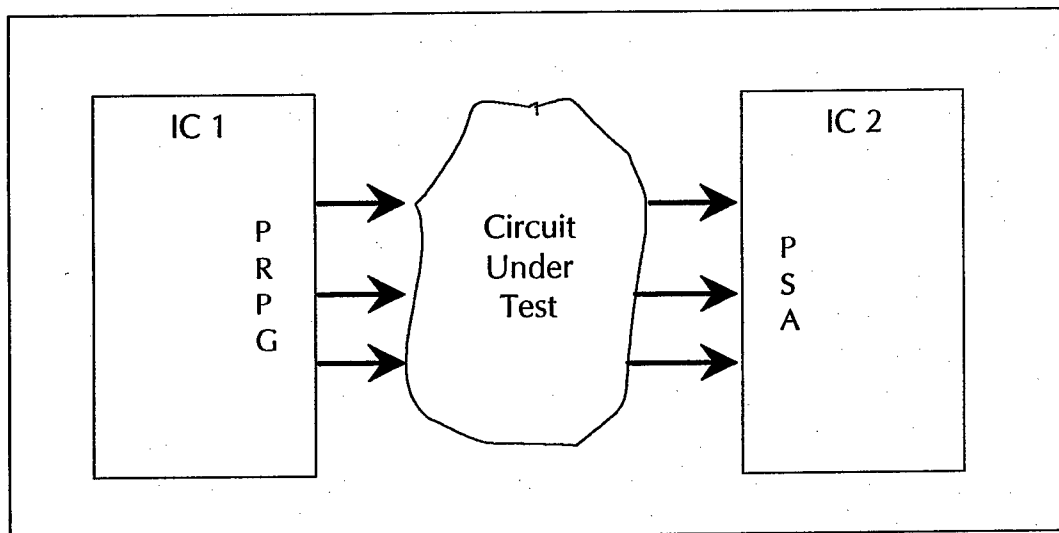


Figure 2.1.2-4. Using Boundary Scan for PWB BIST

Even if only a few such components exist on a board, they can be used to create BIST on that board if well positioned, such as at bus interfaces or interfaces to embedded regular arrays on the board (RAM, ROM, etc). For example, there have been cases where BIST has been implemented on a very complex mixed signal board by simply adding a controller to an ASIC having boundary scan and replacing just a few existing interface parts with BISTed boundary scan interface parts. With the final addition of a few loop-back capabilities at various points and a specialized BIST response evaluator for the analog portion of the test, the board can be tested and diagnosed entirely with BIST, aided by a simple, PC-based tester.

Issues.

If there is no embedded TBC in the system, IC BIST may be initiated in hardware via a memory-mapped interface or by a discrete pin. The IEEE 1149.1 specification states that the TAP must be in RUNTEST/IDLE mode to execute BIST. However, if no TBC is driving the bus, the TAP will remain in the RESET state. To execute BIST, control must be taken from the TAP so the BIST controller can manipulate the test logic. This may be as simple as multiplexing the TAP signals and the BIST controller.

2.1.3. Real-time event qualification.

As semiconductor technologies continue to increase the speed at which ICs operate, at-speed functional testing of boards and systems becomes more difficult. Traditionally, boards are tested at-speed using high performance functional testers and probing fixtures. However, as physical access to electronic circuit assemblies diminish, probing becomes difficult and in some cases impossible. Therefore, a new approach is needed to allow at-speed board testing to be performed by IC resident test logic, instead of external testers and probing mechanisms.

While 1149.1 describes how the Sample/Preload instruction is used for system data sampling, an attempt to use it will quickly reveal problems not addressed in the standard.

One of the problems involves synchronizing the test clock (i.e. the clock that times the test architecture) to the system data traversing the boundary of the IC, so that system data may be sampled in a stable, non-transitioning state. If a system clock synchronous to the data transfers is available, it can be substituted for the test clock by adding clock switching circuitry on the board design to solve this problem.

Another problem is that the system clock switched into the test clock may operate too fast for the 1149.1 standard's test access port (TAP) interface. The upper clock rate for a TAP is technology dependent, but a 15 to 20 megahertz clock is sometimes an upper limit on some logic families. If the system data transfers through the ICs boundary at a rate which exceeds the maximum TAP clock rate, it will be impossible to execute the Sample/Preload instruction because the TAP will be unable to operate.

Still another problem involves qualifying when data is to be sampled. In order to obtain meaningful data, the sample operation should be timed with the occurrence of an expected event. Sampling data synchronously, but at random does not provide any useful test information. By adding qualification and control logic to the board design, the sample operation can be timed with a board level event, making the instruction useful.

From the description of these problems, it is clear that it is a challenge to use the Sample/Preload instruction for at-speed system data sampling. The amount of additional logic that must be implemented on the board to make this instruction useful is impractical in most cases. It is also important to remember that the data sample rate is dependent on the maximum test clock rate of the slowest IC TAP on the board design.

EVENT QUALIFICATION.

An event qualification architecture (EQUAL) has been developed to provide a method of enabling test logic in a IC for the purpose of executing at-speed test operations. While the EQUAL architecture can be used to control a variety of different IC resident test structures, combining it with the 1149.1 boundary scan is ideal for testing at-speed data transfers between ICs on a board. EQUAL's ability to enable boundary test logic to capture at-speed data transfers between ICs allows testing for timing sensitive and/or intermittent failures that may occur during system operation. These types of failures are difficult if not impossible to detect without probing the board design.

EQUAL ARCHITECTURE.

The EQUAL architecture in the IC of Figure 2.1.3-1 consists of a controller, referred to as an event qualification module (EQM), and a series of event qualification cells (EQC), each cell is associated with an IC input/output signal. When enabled, the EQUAL architecture takes control of the IC's boundary test logic and causes it to operate synchronous with the host IC during test. While controlled by the EQUAL architecture, the boundary test logic can perform at-speed data sample or signature analysis test operations concurrent with normal IC operation. After the test is complete, the EQUAL architecture is disabled, allowing the boundary test logic to be controlled by the 1149.1 TAP.

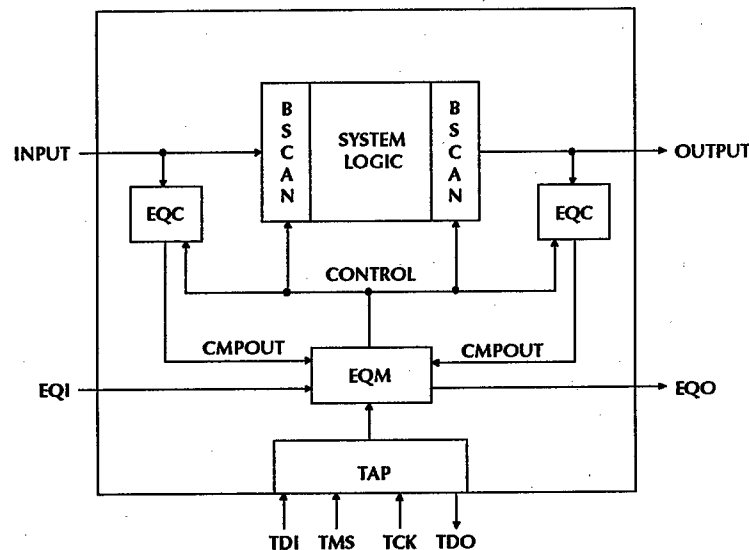


Figure 2.1.3-1. Event Qualification Architecture

Event Qualification Cells.

The EQCs contain compare circuitry and scannable latches for storing start, stop, and mask compare bits. The EQCs have inputs for receiving the boundary signal to be compared and for receiving control from the EQM. The EQCs have a compare output (CMPOUT) to transmit the result of the compare operation to the EQM. When a match occurs between the boundary signal and the start or stop compare bit, the CMPOUT output transmits a match signal to the EQM. The EQC compare circuitry can be masked so that the EQC outputs a match signal on CMPOUT regardless of whether a match occurred between the pin signal and selected compare bit. The EQC's mask bit allows assigning "don't care" conditions to a boundary pin signal that is not required in the qualification of a particular test operation.

Event Qualification Module.

The EQM is a state machine that can be enabled to control test logic in the IC design. The EQM has inputs for receiving the CMPOUT signals from the EQCs, an external event qualification input (EQI) signal, and scan access control signals from the TAP. The EQM has outputs for controlling test logic and EQCs and for outputting an external event qualification output (EQO) signal. While not shown in Figure 2.1.3-1, the EQM receives one or more system clocks from either the internal system logic or from an external system clock input. The system clock input to the EQM is synchronized with the boundary data transfer to allow valid data sampling.

Inside the EQM the CMPOUT inputs are combined into one composite compare signal. By monitoring this composite compare signal the EQM can sense when an expected pattern has occurred at the boundary of the IC to start a test operation. Alternately, the EQM can monitor the EQI input instead of the internal compare signal to start a test operation. The EQI input is used when qualification of a test is not based only on the

local boundary conditions of the target IC, but rather over a range of IC boundary conditions and/or other signals generated on or input to the board design.

EOM TEST PROTOCOLS.

The EQM has eight test protocols that can be selected and used to perform testing. The eight protocols are:

1. Protocol 1 This protocol allows for testing in response to an Nth event, and repeating the operation M times.
2. Protocol 2 This protocol allows for testing during an Nth event, and repeating the operation M times.
3. Protocol 3 This protocol allows testing in response to a first Nth event, stopping storage in response to a second Nth event, and repeating the operation M times.
4. Protocol 4 This protocol allows for testing in response to a first Nth event, stopping storage after N clocks in response to a second Nth event, and repeating the operation M times.
5. Protocol 5 This protocol allows for testing for N clocks in response to an Nth event, and repeating this operation M times.
6. Protocol 6 This protocol allows for pausing for N clocks in response to an Nth event, then testing for N clocks, and repeating this operation M times.
7. Protocol 7 In response to an Nth event, this protocol allows for testing for N clocks, then pausing for N clocks, and repeating the testing and pausing for M-1 times.
8. Protocol 8 In response to an Nth event, this protocol allows pausing for N clocks, then testing for N clocks, and repeating the pausing and testing steps M times.

Protocols 1, 2, and 3 will be described in detail in this section. A protocol 1 operation allows a test operation to occur once in response to an event. A protocol 2 operation allows a test operation to occur while an event is present. A protocol 3 operation allows a test operation to be started in response to a first event and stopped in response to a second event. The test operation referred to in the protocols can be used to control any type of test logic structure residing in the host IC. However, in this section the protocols are used to control an IC's boundary test logic.

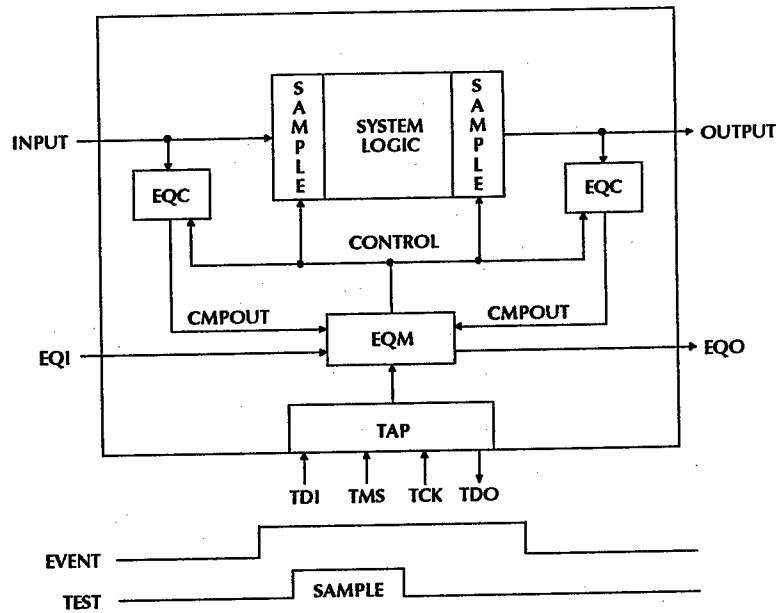


Figure 2.1.3-2. EQUAL Protocol 1

Protocol 1 Description.

In Figure 2.1.3-2, a protocol 1 operation is used to take a snapshot sample of the data passing through the IC's boundary at a qualified point in time. To set up a protocol 1 operation, the EQCs at the IC's boundary are first scanned with an expected boundary pattern. This operation is similar to setting up the qualification conditions on a logic analyzer pod connected to the boundary of the target IC. Since only a single event needs to be detected using this protocol, only the start bit in the EQCs is used. Also if one or more pins on the IC are not required in the event qualification process, the mask bit in their EQCs can be set to force a match output on the CMPOUT signal.

After the expected boundary pattern is loaded, the EQM is scanned to load the protocol 1 command. After the command is loaded the EQUAL architecture becomes armed and begins polling for the expected event. Since the system clock driving the EQM controller is synchronized to the boundary data transfers, each boundary data pattern is compared against the expected data pattern. When a match is detected the EQM controller issues control to the boundary test logic to perform a data sample operation. After the test operation is complete, the sampled data is scanned out for inspection via the 1149.1 TAP.

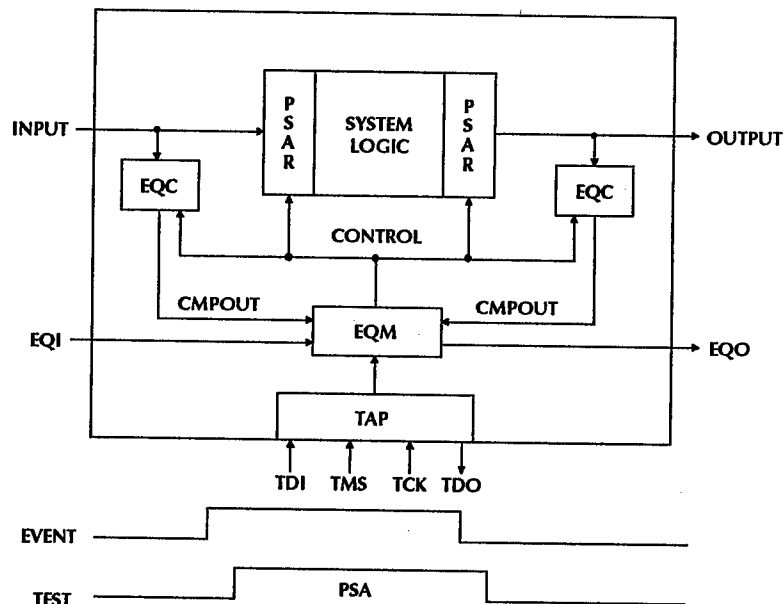


Figure 2.1.3-3. EQUAL Protocol 2

Protocol 2 Description.

In Figure 2.1.3-3, a protocol 2 operation is used to take a signature of the data passing through the IC's boundary while an event is present. During this test operation the boundary test logic is setup to operate as a parallel signature analysis register (PSAR). The setup procedure for this protocol is identical to protocol 1.

After the expected boundary pattern is loaded, the EQM is scanned to load the protocol 2 command. After the command is loaded the EQUAL architecture becomes armed and begins polling for the expected event. When the event is detected the EQM controller issues control to enable the boundary input and output PSARs to operate with the system clock to collect signatures of the data passing through the IC's input and output boundary while the event is present. After the test operation is complete, the signatures are scanned out for inspection via the 1149.1 TAP.

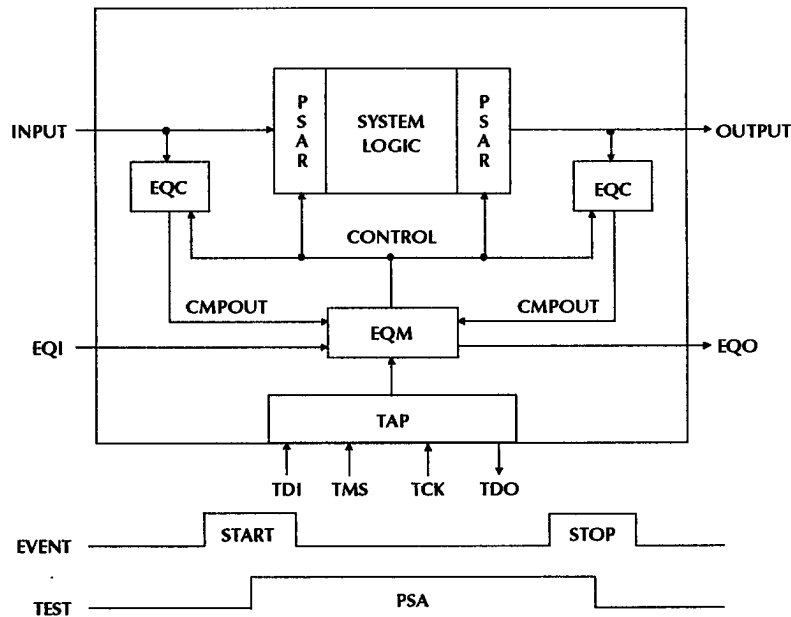


Figure 2.1.3-4. EQUAL Protocol 3

Protocol 3 Description.

In Figure 2.1.3-4, a protocol 3 test is used to take a signature of the data passing through the ICs boundary over a window of time determined by a start and stop event. During this test operation the boundary test logic operates as a parallel signature analysis register. The setup procedures for this protocol is identical to protocol 1, except that both the start and stop compare bits in each EQC are used to allow comparing against both a start and stop boundary pattern.

After the expected boundary patterns are loaded, the EQM is scanned to load the protocol 3 command. After the command is loaded the EQUAL architecture becomes armed and the EQM outputs control to the EQCs to compare the data passing through the ICs boundary against the expected start pattern. When a match occurs, the EQM outputs control to enable the PSARs to collect signatures and to cause the EQCs to begin comparing for the stop pattern. When the EQM detects the occurrence of the stop pattern, it outputs control to disable the PSARs from collecting signatures. After the test operation is complete, the signatures are scanned out for inspection via the 1149.1 TAP.

GLOBAL EVENT QUALIFICATION.

While local IC qualification, as previously described, serves many testing needs, there are times when the qualification of a test needs to be expanded beyond the boundary of the target IC. Increasing the number of boundary signals participating in the qualification process improves the resolution as to when a test operation is enabled. For example, the signals at one ICs boundary may not provide sufficient qualification for a particular test operation. However, by combining the boundary signals of neighboring ICs with the boundary signals of the target IC, a global qualification mode is obtained which can be used to more accurately enable the test operation.

In Figure 2.1.3-5, a global qualification example is shown. During global qualification the EQO signal from each IC is set to output the result of the IC's local boundary compare operation. The EQO signals from all the ICs are input to the voting circuit so they can be combined into one composite global compare signal. Any IC that does not participate in the global qualification sets its EQO output to a state that will not interfere with the operation of the voting circuit. The output of the voting circuit is fed back into each IC in the circuit, via the EQI signal, to allow the EQMs to monitor the global compare signal.

Each time a match occurs across all the IC boundaries, the voting circuit outputs a global compare signal back to each IC. The target IC(s) respond to the global compare signal to execute a predetermined test operation. If a protocol 1 or 2 type qualification is used, the voting circuit outputs a single signal to enable a test operation. If a protocol 3 type qualification is used, the voting circuit outputs a first signal to start a test operation and a second signal to stop the test operation. By monitoring the event (EVT) output the 1149.1 test bus controller can determine when the global test operation is complete so that the test results can be accessed via scan.

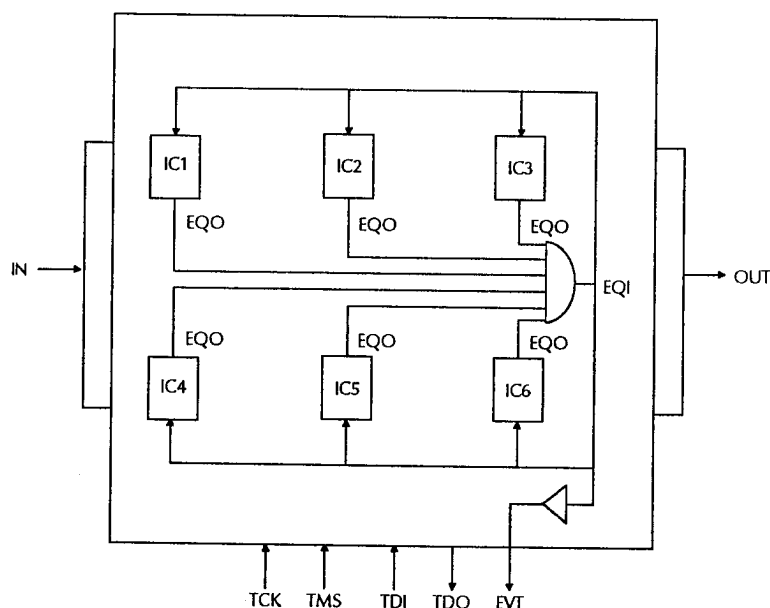


Figure 2.1.3-5. Global Event Qualification

EQUAL APPLICATIONS.

The ability to non-intrusively monitor an operating circuit or system using only the 1149.1 test bus and an interrupt as interface, has many attractive applications. One benefit of this approach is that the test equipment requirement in many instances can be met with a low cost, portable computer with software tools for accessing 1149.1 compatible components. The following applications illustrate some of the uses of this approach in different areas of testing.

Functional Testing Applications.

Testing system logic for at-speed functionality can be very challenging. Blending a traditional functional test approach in with the ability to utilize an IC's boundary test logic to collect at-speed boundary signatures provides an alternative method of diagnosing functional failures. For example, if a board designed with ICs implementing the EQUAL approach fails a functional test, the boundary signatures of each IC can be scanned out and compared to expected values. By determining which boundary signature(s) failed, the fault can be traced back to one or more of the ICs on the board design. Sometimes the boundary signatures may reveal an error condition not detected by the functional test. In this case the technique can be used to improve the fault coverage of a functional test.

System Integration Applications.

During system integration, hardware and software sections of a system design are merged together to complete the last phase of development. The typical approach used during system integration is to utilize external logic analyzers and other test instruments that can view the at-speed operation of the system. While the EQUAL approach cannot completely eliminate the need for external test equipment it can provide improved visibility into the internal components of a system. The EQUAL approach can be used to enable data sample or signature analysis operations at potentially all the IC boundaries in a system. Having such an improved view of the system can play a key role in reducing the amount of time and cost required for system integration and debug.

Environmental Testing Applications.

Environmental testing is performed on most high end commercial and military systems. This test is designed to prove that the system operates properly in all anticipated environments. Usually the system under test is sealed and the tester can only monitor its primary outputs. If a system fails in a particular environment it is difficult to duplicate the cause of the failure when the system is dismantled and retested outside the chamber. By using EQUAL architecture, at-speed boundary information can be obtained from each IC in the system while it is in the environmental chamber. Thus an environmental related failure can be more easily detected and diagnosed, improving the quality and reliability of the system.

2.1.4 Embedded Software Emulation and Debug via IEEE 1149.1

The IEEE 1149.1 standard lays the foundation to new approaches to software/hardware integration and debug for systems implementing embedded processors.

Traditional emulation approaches.

Traditional approaches for debugging systems with embedded processors utilize a technique called 'emulation', which involves the physical substitution of an embedded processor with a substitute test facility (See Figure 2.1.4-1). The common name for this is In-Circuit-Emulation (ICE). Using an ICE provides the low-level control of the system which is required for software/hardware integration and debug.

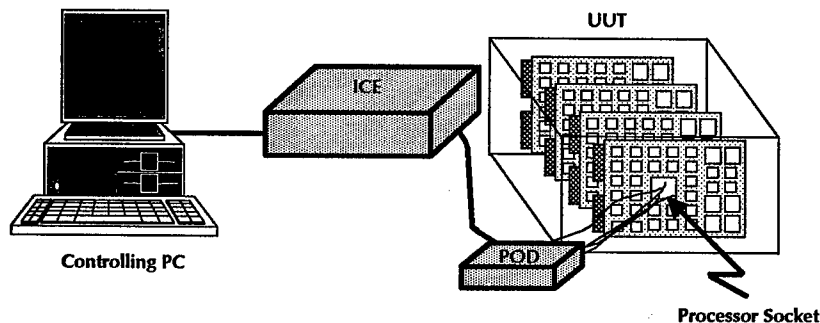


Figure 2.1.4-1. PC Based Test Facility

The use of an ICE requires the replacement of the processor with a large interface pod. This pod provides the timing and event qualifications required for software and hardware debugging of the target. The following features are usually provided:

- Software breakpoints - Stopping on program instructions within the RAM memory space. Accomplished by instruction replacement.
- Memory breakpoints - Stopping on accesses to/from the processor's memory space (RAM, ROM).
- Hardware breakpoints - Stopping on processor events; interrupts, control lines, hardware events or signals.
- Real-time instruction trace.

While the ICE provides many necessary and useful features, it comes at some expense. The most technically imposing feature is that the system is not using the real processor during the debug cycle. After system debug, when the real processor is returned, new problems often arise. These new problems cannot be found using the ICE, since the ICE masks the problem.

Other restrictions imposed by the traditional ICE include:

- The target CPU must be socketed and not conformally coated.
- A large pod must be placed near the system processor socket usually requiring a board extender for physical access.
- The ICE must be in close proximity to the processor, usually within a few feet. This may require hanging the ICE unit over the processor.
- The high cost of the ICE primarily due to the design using high speed memory and timing logic.
- Internal processor must be disabled.

1149.1-based capabilities.

The introduction of the IEEE-1149.1 scan standard provided a new way to approach the problems of system debug. The standardized access provided by IEEE 1149.1 provides an infrastructure for 'Embedded Emulation' that allows debugging features similar to traditional ICEs without the need for processor replacement or board extenders. Instead of processor replacement, the processor functions are controlled and observed via the

1149.1 test bus. The need for a large processor replacement pod and close proximity of the tester has been eliminated. Also, the cost of the debugger is reduced.

Figure 2.1.4-2 illustrates the control mechanism for an embedded emulation capability. In this architecture, access to the process is provided by IEEE 1149.1. Access and control for the 1149.1 test bus is accomplished via a PC-based system which contains 1149.1 control hardware and software. This approach allows a common test controller for the 1149.1 test operations and the processor emulation functions.

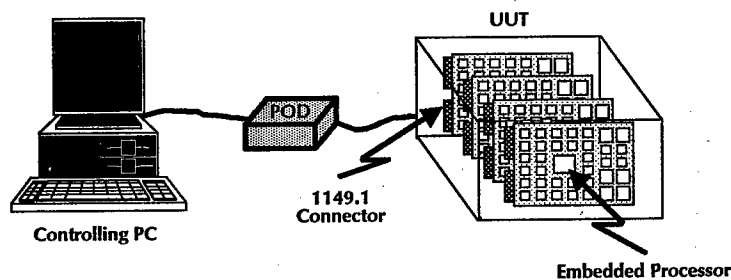


Figure 2.1.4-2. Embedded Emulation Control

Debugging using the actual system processor minimizes cabling, timing and emulation issues and allows for remote testing, such as environmental chamber testing, sub-system testing, or final module testing.

The use of 1149.1 also provides the opportunity to perform simultaneous debugging of multiple processors and access to other test devices. This is becoming very important as systems use more processors.

Scan Based Debuggers.

Scan based debuggers can offer many of the features found in ICE systems. The main limitations come from limited or no internal trace memory. The following features can be embedded in scan based debuggers:

- Read/write memory available to the processor.
- Read/write general purpose processor registers.
- Read/write processor control registers.
- Resetting the processor.
- Starting/stopping the processor.
- Software breakpoints in the RAM memory space.
- Hardware breakpoints, if the processor supports them.
- Internal trace, if supported.
- Cache control.
- Single stepping the processor.
- Easy interface for downloading object code or data to memory.
- Disassembler.
- High-level source code debugging, if object code debug information is present.

- Simple hardware interface board on host debugger computer .
- No target monitor code or other code modifications required.
- No target processor resources required; memory, I/O ports, interrupts.

Figure 2.1.4-3 illustrates potential user interface features for an embedded emulator. From this interface the user can control software execution, examine actual execution steps, and view registers.

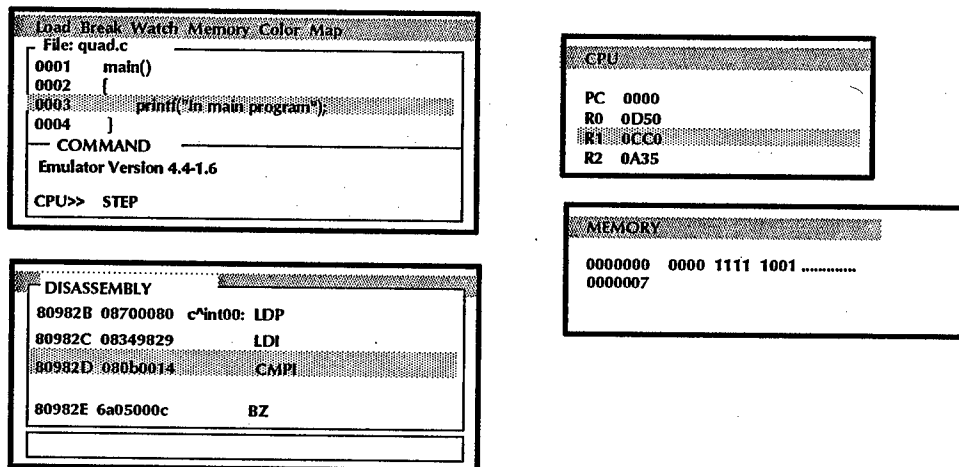


Figure 2.1.4-3. Embedded Emulator User Interface Features

The resolution of functions that can be performed is related to the level of scan implemented. While many emulation functions can be executed on a processor implementing only boundary scan at the pins of the device, a finer level of debug control is possible when internal scan is implemented in the device at strategic locations to control/observe registers, etc. For example, when a hardware device (e.g. TMS320C40) is designed with embedded emulation as a requirement, a powerful support environment can result from the boundary scan, internal scan, and support tools.

The applications of multi-processor modules are becoming more widespread, particularly in digital signal processing applications. As illustrated in Figure 2.1.4-4, the control of a multi-processor module can be effectively achieved through the use of boundary scan and embedded emulation. In this illustration, individual windows can be displayed for individual CPUs; the CPU1 window is stepping through source code and the CPU2 window is displaying disassembled instructions. Global windows control the user interface display and coordination between the embedded emulation features.

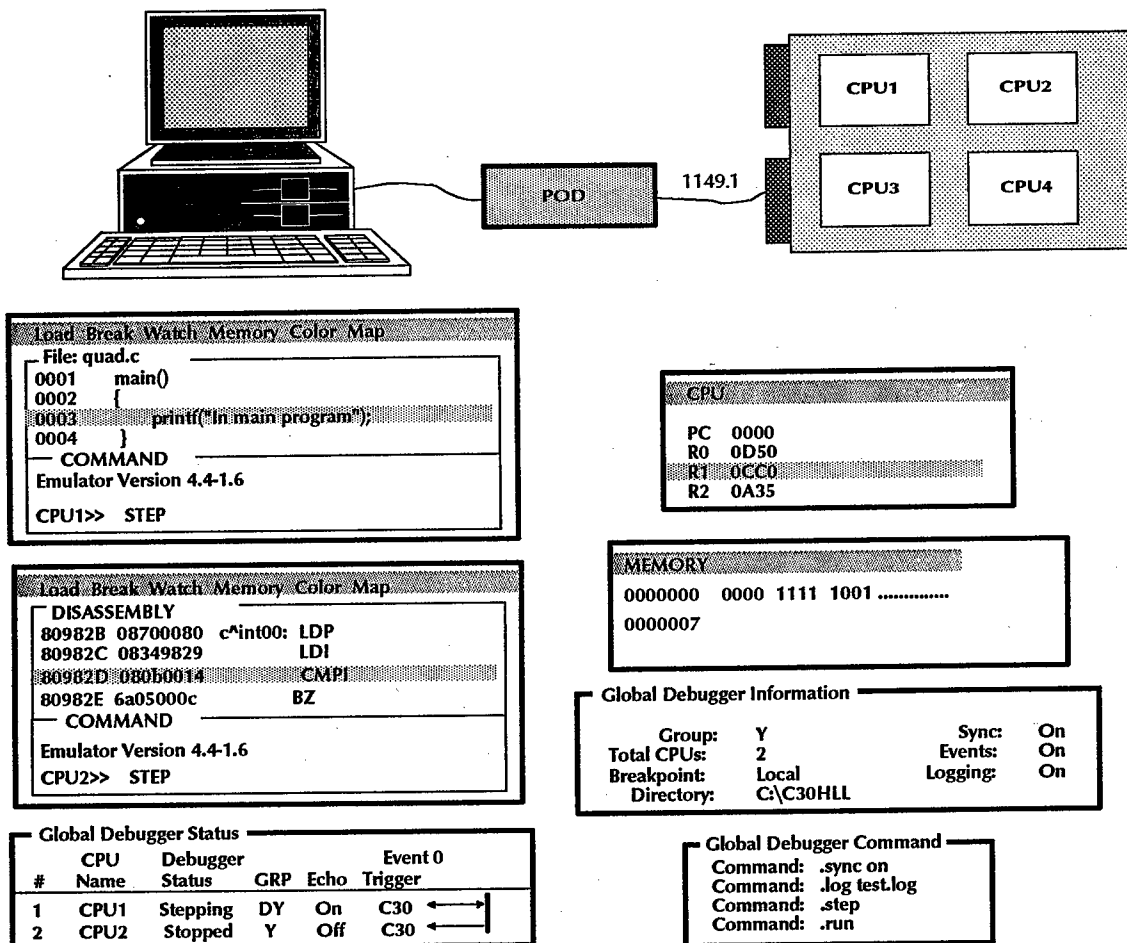


Figure 2.1.4-4. Multi-Processor Embedded Emulation Control

While Figure 2.1.4-4 illustrates an advanced application, more basic applications of boundary scan can achieve significant payback for embedded software/hardware integration and verification. Figure 2.1.4-5 shows an application in which boundary scan devices surround an embedded EPROM in a system. Utilizing the boundary scan devices surrounding the EPROM, the address and control pins of the device can be manipulated and data from the data pins sampled. This simple implementation gives users a great deal of control in executing and observing the system. The user can single step through the EPROM, execute a range of EPROM code, view the EPROM contents, or if the surrounding boundary scan devices have PRPG/PSA capabilities, quickly verify the EPROM contents via CRC.

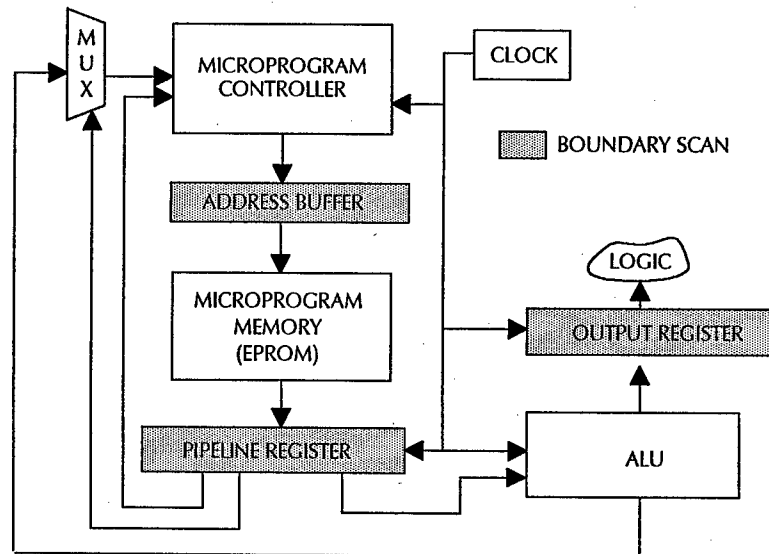


Figure 2.1.4-5. Using Boundary-Scan during System Debug

Scan Limitations.

The use of 1149.1 scan for processor debug can offer many unique features, but it also has some limitations. These limitations are imposed by the separation of the scan clock from the system clock and the serial nature of scan.

Scan-based embedded emulation may not provide the hardware timing and real-time instruction trace features of the traditional ICE. Also it may not allow the breakpointing on data accesses or breakpointing of code in ROM. These limitations can be overcome if the processor contains internal support for hardware breakpoints.

The real-time tracing and timing features are especially useful to the hardware engineer, during system integration. This deficiency can be overcome by using a logic analyzer with scan. In fact the debugging software could provide control of the logic analyzer. The real-time issues may also be solved by adding 1149.1 scannable support chips, such as the Digital Bus Monitor, which contain event qualification features described previously.

2.1.5. Fault Emulation.

Emulation of physical faults is a new approach to testing systems. The IEEE 1149.1 test bus and boundary scan architecture makes this technique even more powerful through the capabilities it provides to control and observe circuitry at the node level. The following paragraphs discuss some of the embedded capabilities which are required for enhanced fault emulation, including the IEEE 1149.1 test bus and boundary scan architecture as well as, design for testability, hardware Built-In Self Test (BIST), and software Built-In Test (BIT).

Fault emulation (or virtual fault insertion) is a viable method to supplement or replace physical fault insertion to verify BIT/Diagnostic software and hardware. Additionally, fault emulation may be useful for applications hardware and software testing and to verify

hardware/software robustness and fault tolerance. Some of the key benefits of fault emulation are:

- The ability to model faults on actual hardware.
- The ability to inject faulty states into systems with no physical modifications or damage.
- The ability to inject faults into systems when physical fault insertion is impossible (i.e., internal silicon, sealed packages ,etc.).

BIT VERIFICATION.

Verification of Built-In-Test (BIT) hardware and software fault detection and isolation capabilities has always been a nontrivial task. In the past, BIT verification has been accomplished through time and labor intensive engineering analysis or via physical fault insertion as outlined in MIL-STD-470 Task 203 (Maintainability Demonstrations). Attempts have also been made to fault simulate BIT, however many designs exceed the capabilities of fault simulators because of design complexity, number of test patterns, or time to execute.

Physical fault insertion may not be an attractive option because it is time consuming and may permanently damage expensive or limited hardware. Additionally, new high density surface mount packaging techniques may make physical fault insertion difficult if not impossible.

As mentioned above, three common methods have been used to verify test effectiveness including, fault simulation, engineering analysis, and physical fault insertion. Each has advantages and disadvantages which will be addressed below. Another key parameter which must be measured is the false alarm rate. False alarms must be measured in real-time under actual operating conditions. None of the three methods above address the issue of false alarm verification.

FAULT SIMULATION.

Fault simulation has been used at the IC level and on individual PWBs using stimulus developed for design verification or PWB ATE tests. This is useful for verifying manufacturing tests, but not for measurement of test stimulus generated by the embedded system BIT. Most systems rely on embedded BIT software to discover faults and identify their location at power-up, on-command, and/or as a background task. A power-up BIT applying 50 million clocks to a 140K gate PWB would only require 5 seconds to execute in real-time. A fault simulator on the other hand, simulating execution of the BIT, may require many months or years to perform a fault simulation with as many clocks.

Even hardware accelerators do not solve execution time problems with very large designs. For example, one complex PWB design was estimated to require over 360 days to fault simulate BIT using a fast hardware accelerated fault simulator. Other issues concerning fault simulation are that all faults are weighted equally, whereas most BIT

requirements are weighted by failure rate, and available simulators lack the ability to provide a PWB fault isolation grade.

ENGINEERING ANALYSIS.

Engineering analysis has been used to both allocate BIT coverage and estimate the effectiveness of BIT. This method can be performed at a high level, for instance at the block or functional level, or at a low level, such as the device or gate level. Low level analysis requires detailed knowledge of both the hardware and of the BIT software and is typically performed by the BIT engineers. The major drawback is the time and labor required to perform an accurate and detailed analysis. This method is also dependent on the engineering expertise and objectivity of the engineer performing the analysis.

PHYSICAL FAULT INSERTION.

Physical fault insertion has been used for many years to verify BIT fault detection and isolation. This is a straightforward but labor intensive task which involves manually inserting faults, such as stuck-at 1, stuck-at 0, and stuck-opens, in the hardware under test to verify whether BIT correctly discovers the fault and identifies its location. The set of faults selected is statistically based on the total number of faults and the failure rate distribution in the system under test. The fault set is also limited to faults which can be physically inserted (at IC pins or PWB nodes). This represents only a small percentage of the total faults which can occur. Cost, time, and physical access limitations further restrict the number of faults which can be inserted. Fault insertion has been performed by lifting or cutting IC pins and PWB etch or by driving a circuit node to VCC or ground. This may permanently damage some components or PWBs and makes this method unattractive in most cases. Figure 2.1.5-1 illustrates some common physical fault insertion methods on a PWB.

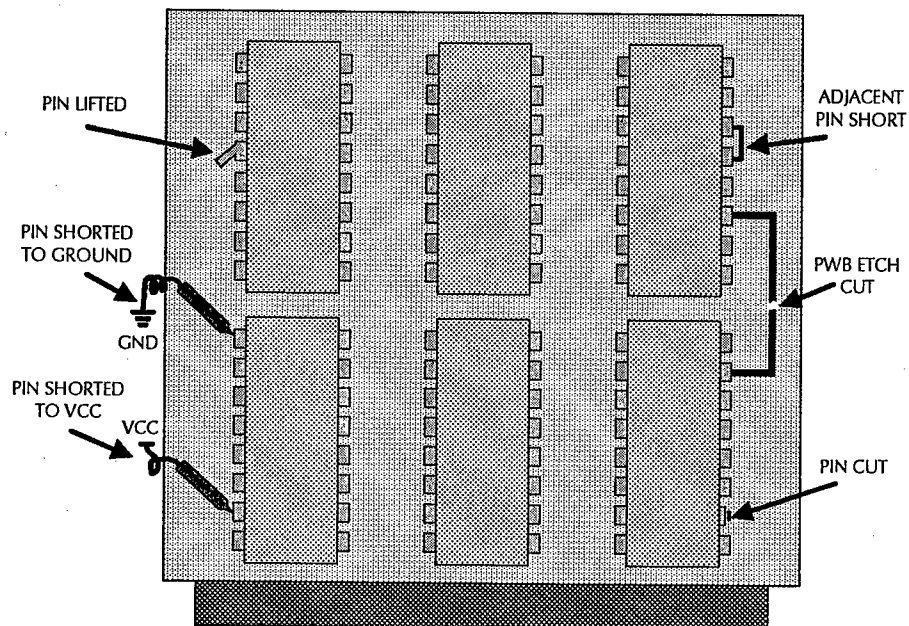


Figure 2.1.5-1. Common Physical Fault Insertion Methods

FAULT EMULATION.

Fault emulation, or virtual fault insertion, is a technique which simulates a faulty state in the hardware under test. This method changes the state of the hardware without the application or BIT software being aware of any perturbation. An analogous scenario would be a computer running a print spooler as a background task while a communications program runs interactively in the foreground. The user and the communications program would not be aware that the print task is executing because it operates independently of the current foreground task. Similarly, fault emulation would allow a faulty state to be introduced while the CPU continues to execute applications software. Fault emulation would also be useful to verify system resistance and response to false alarms by injecting single event upsets into the hardware.

This technique is possible if all or some key portions of the design incorporate internal scan and/or the IEEE 1149.1 test bus and boundary scan architecture. While this technique is not an inherent feature of boundary scan, many systems can be controlled via scan to provide this capability.

FAULT EMULATION VIA BOUNDARY SCAN.

As designs become more dense, physical fault insertion becomes less practical. Some designs are being implemented with hybrid modules and silicon-on-silicon technology. This practically eliminates any possibilities for physical fault insertion. However, with the increased use of the IEEE 1149.1 test bus and boundary scan architecture, much controllability and observability is regained. Via boundary scan and proper design rules, the hardware under test can be controlled such that faulty states can be introduced while BIT software and embedded applications are executing. Hardware can be controlled either through boundary scan of an ASIC or by incorporating ICs with boundary scan in key areas of the design. This level of hardware control is not only useful for fault emulation, but also design verification, integration and testing.

Figure 2.1.5-2 illustrates the basic process of fault emulation. This figure shows a timing diagram consisting of clock and enable signals and a data bus. To insert faulty data in the system the system must first be halted (in this case by putting the clock buffer in boundary scan mode). Next, faulty data is scanned into the scannable data buffers and driven on the data bus. Then the data bus is enabled and clocked which causes the fault data to be latched in the system. Finally, the system is returned to normal operation by releasing the clock signal and allowing the faulty data to be propagated through the system. This method can be used to verify BIT effectiveness, hardware/software robustness and fault tolerance.

For another example, consider a state machine which implements illegal state checking and is buffered by boundary scannable input latches as shown in Figure 2.1.5-3. The fault emulation scheme can be applied step by step as shown in the figure.

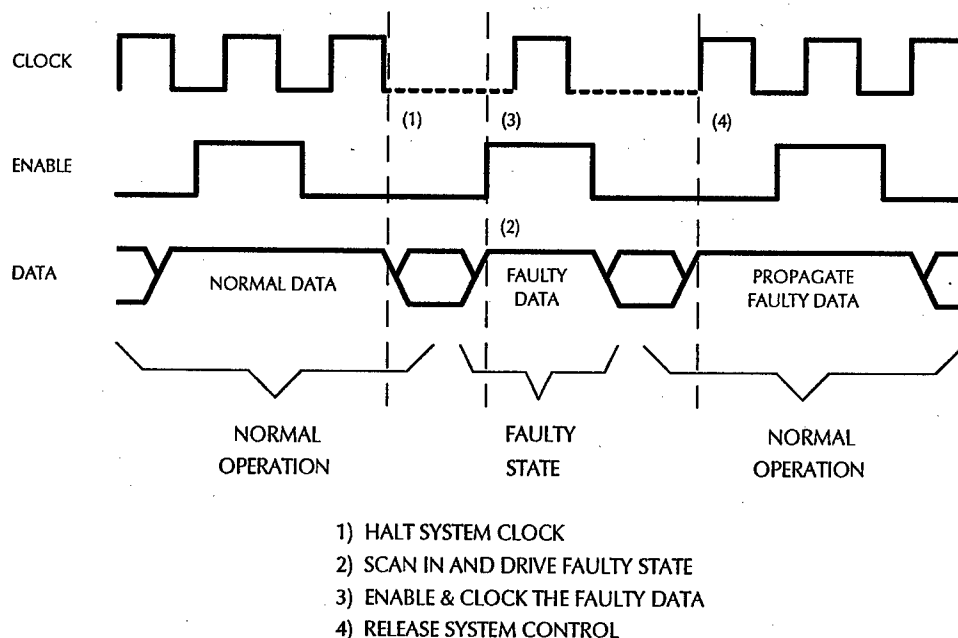


Figure 2.1.5-2. Fault Emulation Timing Diagram

After the fault is injected, the faulty state will propagate through the state machine and, if the error detection function is operating correctly, it will generate an error condition. If scan is also implemented internally, additional errors can be introduced in the internal logic.

For processor based designs, faults can be inserted by controlling the processor via the clock and control lines or by using embedded processor emulation capabilities. Some new processors (i.e., TMS320C50, SuperSparc, etc.) are being introduced with built in emulation features for processor control such as, RUN/STOP, SINGLE-STEP, BREAKPOINT, REGISTER WRITE/READ, etc. While it is definitely easier to perform fault emulation on a processor with embedded emulation, it may not be required.

Consider the design with a conventional processor in Figure 2.1.5-4. If the processor is surrounded with scannable buffers and latches, processor control and data signals can be controlled via the IEEE 1149.1 test bus. The processor can be halted by controlling the WAIT signal and single stepped by toggling the clock. Invalid states can be introduced via the boundary scannable buffers and latches to change memory and I/O values, create invalid control signal states, and generate invalid opcodes and data.

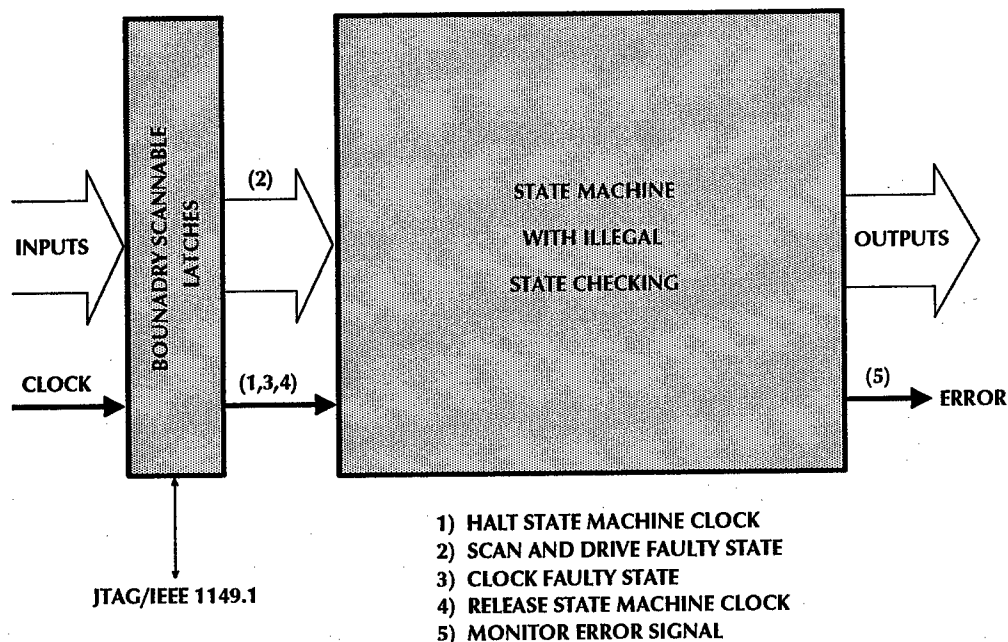
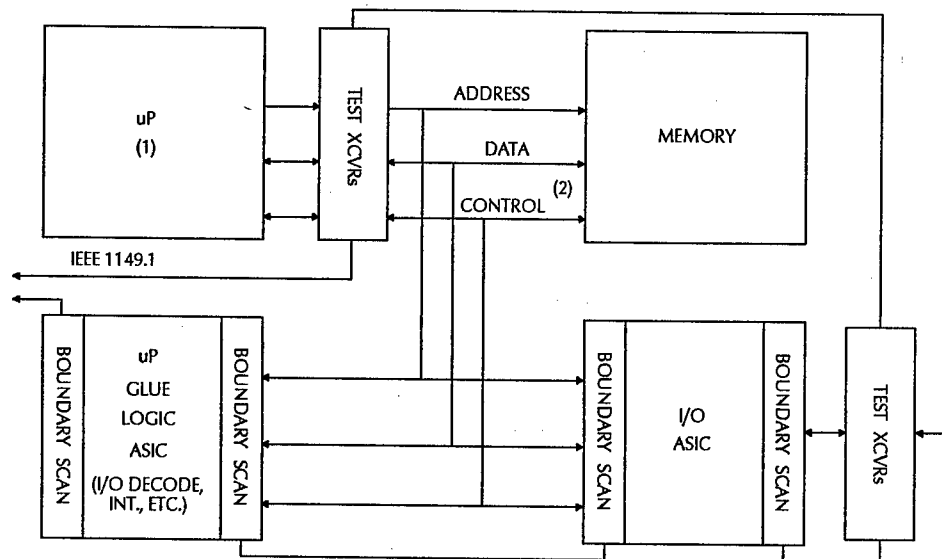


Figure 2.1.5-3. State Machine with Error Checking and Boundary Scannable Inputs

As an example, consider a processor based BIT executing a periodic I/O status test. In this test, if an error condition occurs, appropriate status bits are set in the I/O register. A fault can be introduced by performing the following steps:

1. Halt the processor by setting the WAIT line.
2. Scan the I/O address and faulty data into the scannable devices.
3. Write the faulty data byte to the I/O register.
4. Release the WAIT line.

When the processor executes the I/O status test it will read the faulty status and execute the appropriate error routine. Other faults can be inserted in this way including, RAM errors, illegal instructions, erroneous data, invalid control signals, etc. This technique can similarly be applied to applications software.



- 1) - THE PROCESSOR MAY BE CONTROLLABLE VIA SCAN EVEN IF IT DOES NOT IMPLEMENT BOUNDARY SCAN
 2) - KEY CONTROL SIGNALS INCLUDE: CLOCK, R/W, WAIT, ETC.

Figure 2.1.5-4. Processor Based Design to Support Fault Emulation via Boundary Scan

FAULT EMULATION REQUIREMENTS/LIMITATIONS.

Fault emulation is applicable to many designs. As systems become more integrated and implement boundary scan, fault emulation will become much more valuable for system test, debug and verification. However, there are several primary requirements in order to apply this technique:

1. The circuit under test must have boundary (or internal) scan logic to control the BIT hardware.
2. The circuit under test must have boundary (or internal) scan logic to control the logic to be faulted.
3. The circuit logic must be capable of maintaining it's state with a static clock.

FAULT EMULATION CONTROL.

Fault emulation should be accomplished by manipulating the hardware under test using one controller function to manage the scan state and control the low level bit manipulations. To accomplish this task, a scan-based control system should allow easy and logical control of the boundary and internal scan paths. Experiments have been performed using a TI developed scan-based hardware and software system known as ASSET(tm). The tool maintains knowledge of the complete scan path and allows scannable circuit signals and buses to be treated in an object oriented fashion. In this way, signals such as a processor data bus can be referred to as "PROC_DATA_BUS" rather than the test ICs which drive it (i.e., U12, U32). This provides a level of

abstraction so the hardware under test can be dealt with at a higher and more understandable level.

Simple experiments have been performed which indicated that fault emulation is feasible. Depending on the design for test capabilities of the system under test, various levels of virtual fault insertion are possible. If the system contains internal scan, which is the case on most new designs, internal IC faults can be injected. This allows fault insertion to be performed at a much lower level than physical fault insertion or even fault simulation using behavioral models.

2.2. Module.

The following paragraphs discuss module level test bus extensions. Module level test bus extensions may enhance the functionality of the bus or enhance the test capabilities of the system hardware. Some module level extensions may utilize or leverage IC test bus resources.

2.2.1. Intermodule testing.

Intermodule (module-to-module) testing is obviously desirable either to verify I/O interface logic and module interconnects or to isolate faults to replaceable units for repair or replacement. As systems become more integrated and the military moves to two-level maintenance, module fault isolation will become even more important. A test bus may be used as a means to initiate intermodule tests, coordinate tests, or report test status. There are several methods to implement intermodule testing, some of which include, functional I/O tests, boundary scan tests, and custom I/O test logic. These methods are shown in Figure 2.2.1-1.

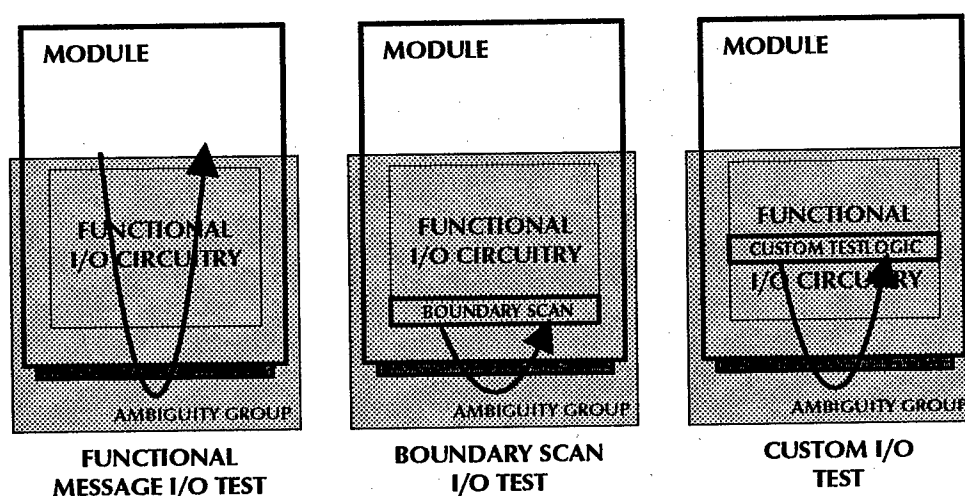


Figure 2.2.1-1. Module Interconnect Test Methods

Functional I/O tests have been and continue to be the most common form of interconnect test. This involves some form of communication (i.e., write, read, etc.) between one module as the bus master and one or more modules as bus slaves. Functional I/O tests,

such as simple memory writes and reads, are easily performed on industry standard backplane buses such as the VME-Bus and custom memory mapped I/O backplane buses. They are popular because the test algorithms are typically simple, they require no test logic, and there are usually multiple destinations on the backplane to communicate with.

The largest disadvantage of functional I/O tests comes from the fact that much of the functional logic must be working to successfully perform a test. If the interface to the functional I/O circuitry is faulty or the functional I/O circuitry itself is faulty the test will fail with fault isolation ambiguity. The fault isolation ambiguity group will depend on the amount of circuitry between the tests source and destination. In many systems, particularly older systems, this is a significant amount of circuitry. Therefore, the less circuitry involved in the I/O test, the smaller the fault ambiguity group.

Boundary scan at the I/O interface offers the best method to detect and isolate faults on modules external interfaces. This is due to the fact that the test patterns can be independently driven from a device's pins without the need to propagate patterns through functional circuitry. The boundary scan circuitry could be controlled via the IEEE 1149.1 test bus driven by a system level embedded test controller or an external controller. Unfortunately, the 1149.1 test bus is not suitable as a module to module test bus because it is a ring topology. As a solution, a module level test bus such as the TM-Bus could be used to control the 1149.1 test bus during intermodule tests. The system level test controller uses the TM-Bus to send commands to drive module outputs on one module and capture module inputs. This operation can be performed via a TM-Bus-to-1149.1 gateway on the module or via the TM-Bus Module I/O Control and Test (MICT) commands, if supported. Alternately, an autonomous test controlled, initiated by a TM-Bus command could locally control the module boundary scan logic during intermodule tests.

Custom intermodule test circuitry is another option which can be used to perform module-to-module tests. This method is implemented by designing specific test circuitry near the module interfaces to drive outputs and sample inputs during test mode. In some cases portions of the functional I/O circuitry may be used during the test and therefore may not offer as good fault isolation as boundary scan. Also, because this method is interface specific, the design may not be reusable on different module types or other systems. However, if implemented, the custom test circuitry could be controller via a module level test bus to provide a generic means of test initiation, coordination, and results retrieval.

2.2.2. TM-Bus with TSMD.

In many applications, it is useful to interface the functionality of a Time-Stress Measurement Device (TSMD) with the Test and Maintenance Bus. This is done to provide temporal environmental data along with performance and failure/maintenance data. Collection of TSMD data supports correlation of real-time performance and failure data with the associated environmental conditions. This is expected to be useful in eliminating false alarms, cannot duplicates (CND's), and Re-Test OK's (RTOKs). It is also expected to support the accelerated maturation of Test and Diagnostic systems.

Figure 2.2.2-1 shows an example of a fault occurrence which is being compared with stress data for correlation evaluation.

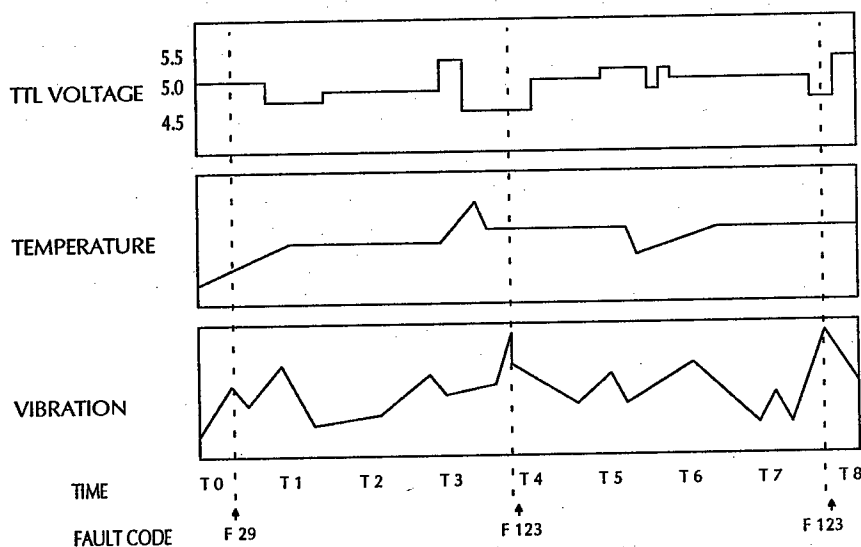


Figure 2.2.2-1. Fault Occurrence Correlation To Stress Data

Access to stress data can be accomplished via the TM bus in several different ways. One of the easiest is to read the stress data directly via the TM bus User-defined Ports. A separate port address may be used for access to each stress measurement sensor or each sensor could be accessed indirectly as a subaddress of the port representing stress measurement data in general. In either of these two cases the primary limiting factor would be the rate at which data can be transferred between the TM bus slave port and the bus master and the rate at which data is being sampled. Using the latest JIAWG specifications, the clock rate would be a minimum of 6.25 KHZ with data packets of 17 bits each (16 bits plus one parity bit). If data from sensors was being sampled at a 1 KHZ rate, generating one word (16 bits) of data per sample, and assuming approximately 3 sensors per module, the total data for the module could be retrieved using this method. This is possible if there are no more than two slave modules on the bus (assuming that the bus could provide an effective transfer rate of 6KHZ allowing for protocol overhead). This would also assume that there was no other information (BIT status, fault log data etc) to be communicated via the TM bus. Obviously this method has serious limitations in the general case of retrieving TSMD data.

Another way to deal with TSMD data is to have data processing capability local to the module which incorporates stress measurement. This is expected to be a more common implementation. The stress data would be processed locally, to determine exceedance of maximum stress levels, to evaluate trends, and to perform other appropriate data transformations. In an implementation of this type, the TM bus throughput impacts are significantly reduced, depending primarily on the level of processing and intelligence implemented at the TSMD. The general operation of TSMDs and some implementation details are described below.

Time-Stress Measurement Devices typically measure parameters such as temperature, vibration, acceleration, voltage, and time. The TSMD may also monitor digital test signals (discretes). A TSMD may be a stand-alone device or may be combined with other circuitry such as a digital interface and non-volatile memory for the fault log implemented on a hybrid device. Most TSMDs provide sampling at a pre-determined or programmable rate. They may include A/D conversion, and preprocessing of the measured parameter(s). Some TSMDs provide D/A conversion and analog and digital outputs which may be used as embedded test stimuli sources. In some applications, the TSMD may also provide local control for the Test Bus interface. An intelligent TSMD block diagram is shown in Figure 2.2.2-2. The figure shows sensors interfaced to a signal conditioning function following analog-to-digital conversion. The data processor also interfaces with the on-chip non-volatile memory for BIT/stress data storage, a real-time clock, and communications ports for module BIT operations/data and for external (backplane/system-level) input/output of TSMD/BIT data.

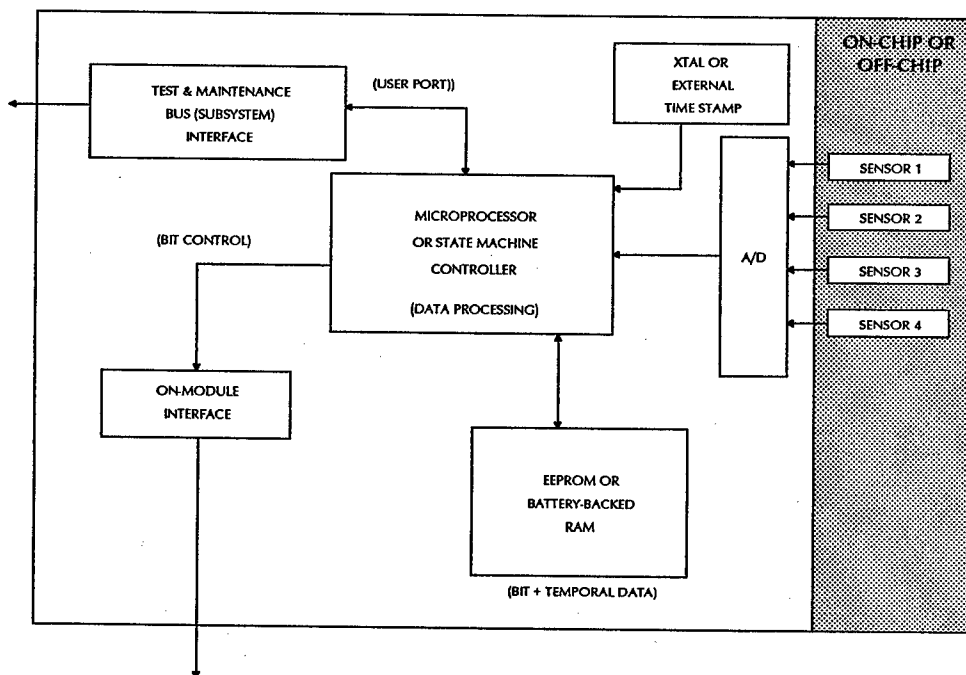


Figure 2.2.2-2. TSMD Architecture

Interfacing the TM bus with an intelligent TSMD may be accomplished in several ways. The interface may be controlled by a state machine, or by an on-chip microprocessor. Although the TM bus protocol can be handled by a state machine, in applications where additional functionality such as IEEE 1149.1 control, BIT/BIST execution control, and signal processing of TSMD data is required, the microprocessor may be an effective trade-off. Decisions related to the selected interface implementation are typically based on whether or not the module already has a microprocessor with available throughput to support BIT and TSMD data processing, and the amount and complexity of data and

operations which must be controlled by the TSMD. In the instances where BIT for a TM bus slave module consists of self-contained, autonomous actions which require minimal control and processing, a state machine or similar control structure may be sufficient.

In a typical subsystem, the TSMD/TM bus device might be implemented with both a microprocessor-controlled TSMD/TM bus interface as the subsystem master, and a state machine controlled TSMD/TM bus for subsequent 'slave' modules. This allows the microprocessor-based TSMD/TM-bus controller to serve as the subsystem CPU for test functions such as BIT and diagnostics control, TSMD data processing, and failure data filtering and processing.

SUMMARY.

Interfacing the TM bus with Time-Stress Measurement Devices may be accomplished through the user-defined ports with direct access to the stress data, or by accessing data which is pre-processed locally. The implementation of the TM bus slave and the TSMD may be accomplished in a single (hybrid) device or with multiple IC's.

The TM bus may be used to initiate or evaluate testing controlled or interfaced through the TSMD, to collect test result data stored on the TSMD, or to retrieve stress measurements from the TSMD.

Key trade-offs in interfacing TSMDs and the TM bus involve the evaluation of TM bus and TSMD data processing requirements. The objective is to determine whether the stress data can be accessed and transferred directly, whether a state machine can handle the pre-processing operations required or if a local processor can provide the required processing capacity for TSMD data, or whether an additional microprocessor is to be added to support TSMD/TM bus/BIT requirements.

For systems where Cannot Duplicates, False Alarms and Retest OK's are important considerations, TSMD's may prove very useful. Their use should be considered whenever accurate diagnostics are critical to mission or safety. Interfacing with TSMDs through the TM bus is straightforward, especially when dealing with an intelligent TSMD implementation.

2.2.3. Extending IEEE 1149.1 in a Backplane Environment.

While 1149.1 was developed to serially access ICs on a board, it can be used at the backplane level to serially access boards. 1149.1 has two serial access configurations, referred to as "ring" and "star", that can be used at the backplane level. The following describes both configurations and identifies problems with each when used at the backplane level.

1149.1 Backplane Ring Configuration.

In a backplane 1149.1 ring configuration, all boards directly receive the TCK and TMS control outputs from a test bus controller (TBC) and are daisy chained between the TBC's TDO output and TDI input. During scan operation, the TBC outputs control on TMS and TCK to scan data through all boards in the backplane, via its TDO and TDI bus connections. The problem associated with the ring configuration, is that the scan operation only works if all the boards are included in the backplane and are operable to

scan data from their TDI input to TDO output. If one of the boards is removed or has a fault, the TBC will be unable to scan data through the backplane. Since the ring configuration does not allow access to remaining boards when one is removed or disabled, it does not fully meet the needs of a backplane serial bus.

1149.1 Backplane Star Configuration.

In a backplane 1149.1 star configuration, all boards directly receive the TCK and TDI signals from the TBC and output a TDO signal to the TBC. Also each board receives a unique TMS signal from the TBC. In the star configuration only one board is enabled at a time to be serially accessed by the TBC. When a board is enabled, the TMS signal associated with that board will be active while all other TMS signals are inactive. The problem with the star configuration is that each board requires its own TMS signal. In a backplane with 50 boards, the TBC would have to have 50 individually controllable TMS signals, and the backplane would have to have traces for each of the 50 TMS signals. Due to these requirements, star configurations are typically not considered for backplane applications.

A NEW BACKPLANE ACCESS APPROACH.

The new backplane access approach described in this section provides a method of using the 1149.1 bus at the backplane level. Using this approach, it is envisioned that one homogeneous serial bus may be used throughout a system design, rather than translating between multiple serial bus types. Employing a common serial bus in system designs can simplify software and hardware engineering efforts, since only an understanding of one bus type is required.

A circuit, referred to as an addressable shadow port (ASP), and a protocol, referred to as a shadow protocol, have been defined to provide a simple method of directly connecting 1149.1 backplane and board buses together. When the 1149.1 backplane bus is in either its run test/idle (RT/IDLE) or TLRST state, the ASP can be enabled, via the shadow protocol, to connect a target board's 1149.1 bus up to the backplane 1149.1 bus. After the shadow protocol has been used to connect the target board and backplane buses together, it is disabled and becomes transparent to the operation of the 1149.1 bus protocol.

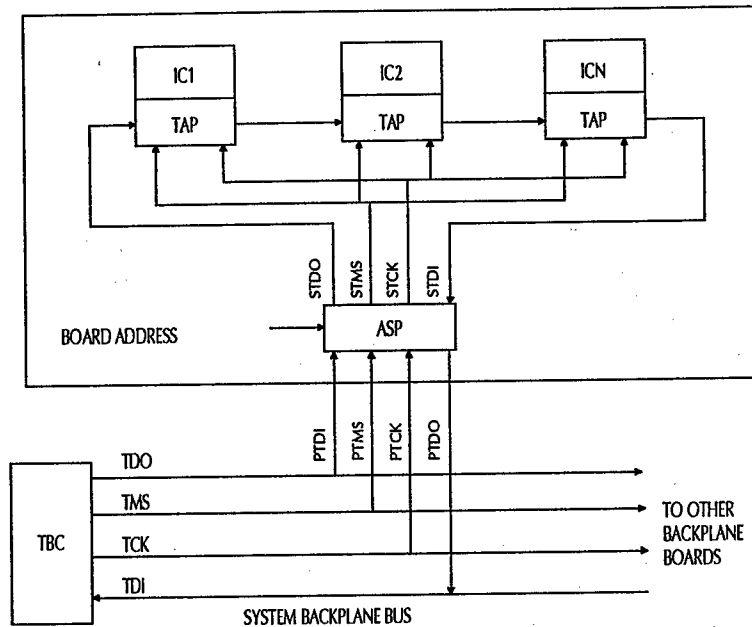


Figure 2.2.3-1. Board Using ASP Circuit

A board example using the ASP is shown in Figure 2.2.3-1. The board consists of multiple ICs and an ASP. The ICs operate, when connected to the 1149.1 backplane bus, via the ASP, as described in the 1149.1 standard. The ASP has a primary port for connection to the backplane 1149.1 bus, a secondary port for connection to the board 1149.1 bus, and an address input. The primary port signals are labeled; PTDI, PTDO, PTCK, and PTMS. The secondary port signals are labeled; STDI, STDO, STCK, and STMS. The address input identifies the board on which the ASP is mounted.

In Figure 2.2.3-2, multiple boards, similar to the one in Figure 2.2.3-1, are shown interfaced to a TBC via ASPs. When one of the boards needs to be accessed, the TBC transmits a selection shadow protocol, referred to as a select protocol, to address and enable the ASP of the selected board. The TBC transmits the select protocol when the backplane 1149.1 bus is inactive and in a stable state, such as the RT/IDLE or TLRST states. The select protocol contains an address that is used to match against the address input to the ASP. All ASPs receive the select protocol, but only the one with the matching address is selected.

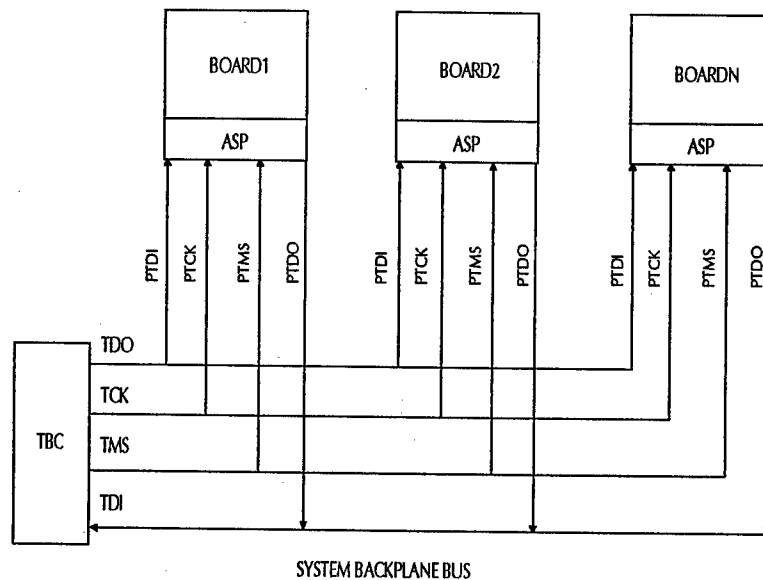


Figure 2.2.3-2. Backplane ASP Connections

In response to the select protocol, the selected ASP transmits an acknowledgment shadow protocol, referred to as an acknowledge protocol, to the TBC to verify reception of the select protocol. The acknowledge protocol contains the address of the selected ASP to allow the TBC to verify the correct ASP was selected. After transmitting the acknowledge protocol, the selected ASP makes a connection between its primary and secondary ports. In response to the acknowledge protocol, the TBC communicates to the selected board using the 1149.1 bus protocol. If the TBC does not receive an acknowledge protocol, it assumes the board has been removed or is disabled and will not attempt to communicate to it using the 1149.1 protocol.

After the TBC completes its 1149.1 access of the currently selected board, it can output a new select protocol to select another board's ASP. In response to the new select protocol, the newly selected ASP transmits an acknowledge protocol back to the TBC, then connects its primary and secondary ports. Also in response to the new select protocol, the previously selected ASP breaks the connection between its primary and secondary ports.

The disconnecting ASP remains in the state the backplane 1149.1 bus was in when the disconnect occurs, i.e. the 1149.1 RT/IDLE or TLRST state. The ability to disconnect and leave a board level 1149.1 bus in the RT/IDLE state is very important since it allows leaving a board in a test mode while other boards are being selected and accessed.

A key objective of this backplane access approach was designing the select and acknowledge protocols so that they could be transmitted on the 1149.1 bus without infringing upon the 1149.1 bus protocol. This objective was met by specifying that the select and acknowledge protocols could not use the 1149.1 TMS signal, and that the protocols could only be transmitted while the 1149.1 bus is idle in its RT/IDLE state or reset in its TLRST state.

To transmit the select and acknowledge protocols without using the TMS control signal, a bit-pair signaling method was designed to allow control and data to be transmitted together on a single wiring channel. During select protocols, the bit-pair signaling method allows the TBC to transmit control and data from its TDO output to the ASP's PTDI input. During acknowledge protocols, the bit-pair signaling method allows the selected ASP to transmit control and data from its PTDO output to the TBC's TDI input. Both protocols include control to indicate: an idle condition, a start data transfer condition, and a stop data transfer condition. In addition, both protocols include a method of transmitting data during the interval between the start and stop data transfer conditions.

The bit-pair signals are output from the transmitting device (TBC or ASP) on the falling edge of the TCK and input to the receiving device (TBC or ASP) on the rising edge of the TCK. Since this timing is consistent with 1149.1 timing, upgrading a TBC to support this approach is simply a matter of forcing the TMS output to hold its present state ("0" for RT/IDLE and "1" for TLRST) while using normal 1149.1 scan operations to transmit and receive the select and acknowledge protocols. The simplicity of this approach makes it an attractive addition to the 1149.1 test bus.

FRAMING OF SELECT/ACKNOWLEDGE PROTOCOLS.

A diagram of the select and acknowledge protocols being transmitted while the 1149.1 bus is in its RT/IDLE state is shown in Figure 2.2.3-3. The T signals shown in the protocol sequence indicate when the TDO to PTDI and PTDO to TDI wiring channels are tristate and pulled high. The first sequence framed between the first and second I signals is the select protocol output from the TBC to the ASP (TDO to PTDI). The second sequence framed between the first and second I signals is the acknowledge protocol output from the selected ASP to the TBC (PTDO to TDI). The select protocol always precedes the acknowledge protocol as shown in the diagram.

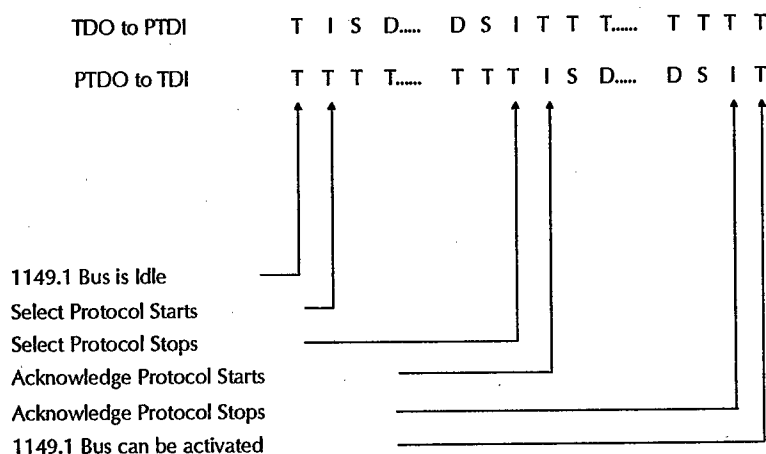


Figure 2.2.3-3. Select and Acknowledge Protocols

The I signal at the beginning of each protocol is designed to be indistinguishable from the preceding T signals. This avoids unintentional entry into a select or acknowledge protocol when the 1149.1 bus enters the RT/IDLE state after a scan operation. However, the I signal at the end of each protocol is designed to be distinguishable from the preceding S and D signals so that it can be used to terminate the protocol. Inside each protocol, first and second S signals are used to frame the address which is defined by a series of D signals. The logic zero and one D signals are distinguishable so that the binary address can be recovered.

ASP CIRCUIT DESCRIPTION.

A circuit example of the ASP is shown in Figure 2.2.3-4. The ASP consists of a receiver circuit (RCR), a transmitter circuit (XMT), a slave control circuit, multiplexers (MX1 and MX2), a power up reset circuit (PRST), and a reset address (RSTA). The primary port signals (PTDI, PTMS, PTCK, PTDO) connect to the backplane level 1149.1 bus. The secondary port signals (STDO, STMS, STCK, STDI) connect to the board level 1149.1 bus. The address input bus receives the board address.

ASP Receiver Circuit

The receiver circuit consists of a controller and a serial input/parallel output (SIPO) register. The receiver's controller determines when a first "I-S-D" signal sequence occurs on PTDI, indicating the start of a select protocol and address input. In response to this input sequence, the controller enables the SIPO to receive the serial address input on PTDI. The controller determines when a first "D-S-I" signal sequence occurs on PTDI, indicating the end of the address input and select protocol. In response to this input sequence, the controller signals the slave control circuit, via the status bus, to read the address on the address input bus (AI), then terminates the select protocol input operation.

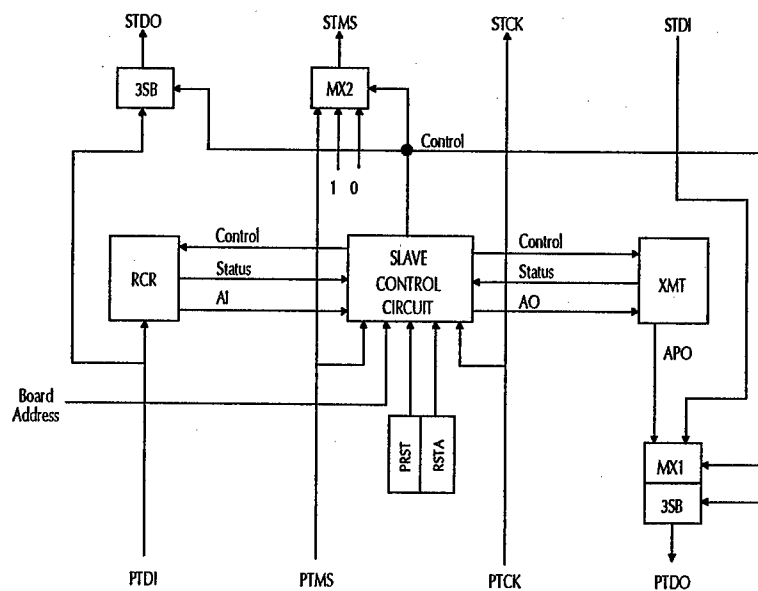


Figure 2.2.3-4. ASP Circuit Example

ASP Transmitter Circuit

The transmitter circuit consists of a controller and a parallel input/serial output (PISO) register. At the beginning of an acknowledge protocol, the slave control circuit enables MX1 and the 3-state buffer (3SB) to pass the acknowledge protocol output signal (APO) from the transmitter to the PTDO output. The slave control circuit then inputs the board address to the transmitter via the address output bus (AO). In response to the address input, the transmitter outputs an I and S signal on PTDO to start the acknowledge protocol, then transmits the address on PTDO. After the address is shifted out, the transmitter circuit outputs an S and I signal sequence to stop the acknowledge protocol.

ASP Slave Control Circuit

The slave control circuit regulates the operation of the transmitter, receiver, and multiplexers during select and acknowledge protocols. During select protocols, the slave control circuit receives parallel address input from the receiver via the AI bus. The slave control circuit compares the received address against the board address. If the addresses match, the ASP responds by outputting an acknowledge protocol. During the acknowledge protocol, the slave control circuit outputs control to the transmitter to load the board address and initiate the acknowledge protocol. After the acknowledge protocol has been transmitted, the slave control circuit outputs control to connect the primary and secondary ports.

Resetting the ASP

When the ASP is reset, the slave control circuit, transmitter and receiver circuits are initialized, and the primary and secondary ports are disconnected. The ASP is reset at power up and the input of a select protocol with an address that matches the reset address (RSTA) inside the ASP. If desired, a reset input pin could also be used to reset the ASP. The RSTA is the same for all ASPs so that a global reset of all ASPs can be achieved by the transmission of a single select protocol containing the reset address. The reset address is unique from the board addresses and is defined to be address zero. An acknowledge protocol is not transmitted after a reset address has been received, to avoid contention on the PTDO outputs of multiple ASPs.

Connecting/Disconnecting ASPs

After access to a board is complete, a new select protocol is issued from the TBC to select the another board's ASP. When the previously selected ASP receives the new select protocol its primary and secondary ports are disconnected and the new ASP's primary and secondary ports are connected. If the new select protocol was issued while the backplane 1149.1 bus was in its RT/IDLE state (PTMS=0), MX2 of the disconnecting ASP outputs a logic zero on STMS, to force the board level 1149.1 bus to remain in the RT/IDLE state. If the new select protocol was issued while the backplane 1149.1 bus was in its TLRST state (PTMS=1), MX2 of the disconnecting ASP outputs a logic one on STMS, to force the board level 1149.1 bus to remain in the TLRST state. The ability to maintain the RT/IDLE state on a disconnected board is very important because it allows tests to be setup and executed on more than one board at a time.

SUMMARY OF ASP ADVANTAGES.

The following is a summary of the advantages the ASP approach offers over the other 1149.1 backplane to board access approaches mentioned in this discussion.

1. The ASP approach enables access to remaining backplane boards when one or more are removed or disabled due to a fault. Thus the ASP overcomes the problem related with 1149.1 ring configurations.
2. The ASP approach does not require use of additional TMS signals to access individual boards. Thus the ASP overcomes the problem related with 1149.1 star configurations.
3. The ASP approach does not require use of sophisticated and bandwidth reducing translation circuitry. Thus the ASP overcomes the problems stated with using different backplane buses to access 1149.1 board environments.

2.3 System.

The results from this test bus evaluation study indicate that there is no need for any new dedicated system level test bus. The current and proposed system buses, used in support of testing applications, are sufficient for supporting the projected weight and throughput requirements in military applications. A study, performed for the Aviation Applied Technology Directorate (AATD), US ARMY Aviation Systems Command, supports this conclusion.

The AATD study evaluated the weight, processing, and throughput requirements for typical avionic subsystems and the capability of system buses to support these requirements. The test bus evaluation study covered similar system buses identified within the AATD study. Most important within the AATD study was system bus throughput requirements. Table 2.3-1 list those system buses studied in the AATD study and their associated throughput requirements. Each bus utilized a bus structure which employed dual bus channels. Table 2.3-2 identifies system bus throughput expansion capability associated with each bus channel. As indicated in Table 2.3-2, minimum usage of each bus was required. The AATD study did not cover the IEEE 488 and IEEE P1394 buses. However, the proposed IEEE P1394 bus is similar to the HSDB in data rate requirements.

Table 2.3-1. System Bus Throughput Requirements

Subsystem	Data Rate	System Bus
Target Acquisition	195.36 KBPS	1553 Chns. 1 & 2
Flight Control	96.00 KBPS	HSDB Chns. 1 & 2
Navigation	195.99 KBPS	1553 Chns. 1 & 2
Weapons & Fire Control	5.00 KBPS	1553 Chns. 1 & 2
A/C Survivability Equipment	256.80 KBPS	HSDB Chns. 1 & 2
	16.00 KBPS	1553 Chns. 1 & 2
Airframe Management	1200.00 KBPS	HSDB Chns. 1 & 2
	23.20 KBPS	1553 Chns. 1 & 2

Table 2.3-2. System Bus Expansion Capability

System Bus	Reserve Capability
HSDB Channel 1	98.5 %
HSDB Channel 2	98.7 %
1553 Channel 1	78.0 %
1553 Channel 2	79.0 %

3. TEST BUS CONTROL

3.1. Introduction.

Test bus control will naturally require some amount of hardware and/or software to perform useful operations. Typical test bus control operations include, test initiation, status monitoring, application and/or capture of test data, verification of faults, and general data communications. Depending on the desired speed, intelligence, flexibility, or programmability, many combinations of hardware and software controllers are possible. The following paragraphs discuss control hardware and software for common test buses.

3.2. Test Bus Control Hardware.

3.2.1. IEEE 1149.1.

Controlling the 1149.1 test bus can range from extremely simple to very difficult operations, depending on the UUT or desired functions. A simple controller can consist of a parallel I/O port or state-machine driven by pre-ordered serial test patterns. A simple controller of this sort requires some function (a simulator or computer) to generate the serial instruction and data register bits in the order of the scan path before data is applied to the UUT. More complex 1149.1 controllers have been implemented which perform parallel to serial translation and TAP state management.

All 1149.1 control hardware options require memory to store test patterns. The test pattern memory required for a given test can vary, depending on the control hardware, based on pattern application methods, storage methods, and any data compression employed. The data storage required for an 1149.1 test is:

$$\text{TEST DATA MEMORY} = ((\text{STIMULUS}) + (\text{EXPECTED}) + (\text{MASK})) + \text{COMMAND DATA} \times \# \text{ OF PATTERNS}$$

This can result in a significant amount of memory. For instance, ignoring command data, a test consisting of 10000 patterns applied to a 64 bit scan path requires: $3 \times (10000 \times 64) = 240,000$ bytes. It is obvious that the number of test patterns should be minimized to keep test data memory requirements as low as possible. As mentioned above, the hardware used to control the IEEE 1149.1 test bus can also vary greatly, ranging from a simple state-machine based controller to a programmable test controller driven by a processor.

On the low end, a simple state-machine based 1149.1 controller can be implemented in hardware using control logic to apply and compare test data stored in a ROM. This implementation is a simple bit-blaster which merely reads ROM data, drives the 1149.1 pins, and compares the expected data to the actual data gated with a compare/don't compare mask. This simple state machine based 1149.1 test controller can be implemented using a Field Programmable Gate Array (FPGA) and ROM to store test patterns. Suitable FPGAs can be found in 28 pin packages, however a 44 pin package may be required for sufficient ROM address and data signals.

Another method to implement 1149.1 test bus control would be via one of the programmable off-the-shelf test bus controller ICs such as the AT&T 497AA boundary scan master, National SCANPSC100 Boundary scan parallel/serial converter, or Texas Instruments 74ACT8990 test bus controller. These devices are mounted in 28 to 44 pin packages each and contain up to 6000 gates. These devices are usually driven by a processor which executes a test program and sequences the test bus controller. A programmable test bus controller would provide a great deal more flexibility than a simple bit state machine-based controller. Programmable modes of operation can be accommodated such as different execution modes, logging modes, and action to take upon predefined or erroneous conditions. This implementation however does require a processor (existing or dedicated) to execute the test program and additional memory (in addition to test data) to store program code.

3.2.2. IEEE P1149.5 (TM-Bus).

The IEEE P1149.5 (and other TM-Bus variants) requires a moderate amount of control logic to implement master and/or slave functions. The TM-Bus logic can amount to 5x to 10x more logic than required for 1149.1 control logic (only counting the TAP, bypass, instruction register, and instruction decode logic). Of course, the TM-Bus logic is only implemented, at most, once per module (as opposed to once per IC). The TM-Bus protocol is also more structured to better suit message passing rather than large bit streams of data. Previous TM-Bus interface ASICs have been implemented which provide dual TM-Bus ports, programmable master/slave TM-Bus operations, a local processor bus interface, a fault log interface, and an interface to an IC level test bus. While all these capabilities are not needed for every application, they are typical of JIAWG-type avionics. An ASIC with the features above requires about 10-12K gates and requires 80-100 pins.

More austere TM-Bus interface implementations would naturally require fewer gates and pins. Many applications only require a module which implements a TM-Bus master-only or TM-Bus slave-only mode of operation. A TM-Bus master-only ASIC, with an interface to a processor bus, would require only about 2000 gates. A TM-Bus slave-only, with gateway to 1149.1 and a processor bus would only require about 4000 gates. These simpler TM-Bus implementations would require as few as 44 pins per device.

3.2.3. MIL-STD-1553B.

MIL-STD-1553B implementations require a significant amount of logic and module area to implement. In the past, several modules were required for the large number of components in the design of the bus. Recently, the number of components has been reduced via VLSI devices and surface mount packaging. Currently, a 1553B interface can be implemented using off-the-shelf components (from companies such as United Technologies or DDC) consisting of a bus controller/remote terminal (BC/RT) ASIC, a transceiver, and a transformer. BC/RT ASICs typically require 84-100 pin packages and the transceiver and transformer requires about an equivalent area.

3.3. Software Control.

Control software for test buses can vary greatly. Factors which determine the simplicity or complexity of test bus control software include the bus complexity, bus topology, system size and diversity, and of course the application.

System and Module Buses.

First realize that module and system level test buses such as the TM-Bus (P1149.5) and MIL-STD-1553B are message based buses. These buses transmit data in message packets to an address on the bus. The only unique information required for common messages to multiple modules is the destination address. This makes software functions fairly straightforward when common message types are used in systems. Also, these buses allow a "broadcast" type message (i.e., run self-test, reset module, etc.) to be transmitted to multiple modules which serves to further reduce bus traffic.

Control software required for the TM-Bus and 1553B bus is application specific. Some systems employ "dumb" interfaces on the buses which simply communicate status or configuration data without any software required. Older systems used this technique to communicate limited data. Newer systems typically employ software which provides some "added value" to the bus in the form of error handling or tolerance, detailed status information, mode command and control, fault logging, etc. In these cases the actual software required to control and communicate via the test bus is small compared to the software used to process the data. Software to perform basic communications (such as a bus master) over the TM-Bus or 1553B bus can be accomplished with less than 2,000 bytes of code.

IC Buses.

Control software for IC buses is extremely application specific. IC buses such as 1149.1 are simple serial data streams, not messages. One "message", which is actually a serial scan, addressed to one IC is completely different if addressed to the same IC type at a different location on the bus. This is due to the fact that the 1149.1 test bus ring addresses ICs by placing the appropriate data at a specified location in the serial stream during each scan. An enormous amount of data (and processing) can be required to perform the bit mapping operations in real-time. Naturally as the size and complexity (number of devices) of the UUT increases, the bit manipulation task becomes more difficult.

Non-deterministic control and generation of 1149.1 serial streams are usually performed off-line by a simulator or in pseudo-real-time by a personal computer or workstation. Pseudo-real-time interactive control and management of the 1149.1 scan path requires a "model" of the UUT scan path on the controlling computer to perform such actions. The scan software and the scan "model" can easily consume several hundred thousand bytes of memory on a computer to perform these operations.

Application of deterministic tests can be performed by the simple 1149.1 control hardware discussed above if the serial scan data is pre-formatted for the UUT. Deterministic data can be captured from a simulator or a pseudo-real-time computer generating the data streams and then embedded in a system to be used for start-up BIT.

Software required for control of 1149.1 deterministic tests can range from 20,000+ bytes of code to implement a processor driven programmable test controller using an off-the-shelf test bus controller, to 0 bytes for a simple "dumb" state-machine based test controller.

3.4. Supporting Control Languages.

3.4.1. Boundary Scan Description Language (BSDL).

When 1149.1 began to see wide acceptance, it became obvious that a common method was needed to describe 1149.1 devices in order to ensure consistent modeling and compliance verification. Hewlett Packard first proposed the Boundary Scan Description Language (BSDL) in 1990 as a way of describing 1149.1 compliant designs. BSDL describes the implementation of 1149.1 in devices so that test logic synthesis and test generation may be automated, as long as each implementation complies with the standard. BSDL extends the use of IEEE 1149.1-1990 by providing a standard format, using IEEE Std 1076-1987 VHDL (VHSIC [Very High Speed Integrated Circuit] Hardware Description Language) syntax, to describe the unique application of the boundary-scan standard within a device. With BSDL, ASIC designers and component vendors can model their specific use of 1149.1 just one time for all purposes. Because people will write BSDL, it must be human-readable. As input to automated tools it must be unambiguously parsable by a computer.

BSDL is meant to describe only those IEEE 1149.1-1990 test elements which vary between devices. Tools geared toward the 1149.1 standard can make use of this language. These tools include testability analysis, test generation, failure diagnosis, compliance monitoring and automated boundary scan synthesis. However, BSDL is not a general purpose HDL (Hardware Description Language). Knowledge of the 1149.1 standard and a BSDL description must be combined to fully comprehend the workings of the test logic. With an expanded VHDL created by boundary scan tools, simulation, verification and synthesis functions can be performed. Support for these capabilities exceeds the scope of BSDL.

BSDL complements the 1149.1 boundary scan standard. Elements mandated by the standard, which never vary between devices, are not included in BSDL. For example, characteristics of the bypass register and the TAP states are fully defined and constant. Including these in BSDL would be redundant and introduce the chance for error.

Applications.

Because BSDL complies 100% with VHDL syntax, it may be included with the device functional description throughout the entire VHDL environment. The real power of BSDL, and where it has the most impact, is to drive dedicated boundary scan tools. In the boundary scan world, one BSDL file would be written for all uses. If the design pinout or selection of 1149.1 bus instructions were to change, only the one BSDL description would be updated. After users change the BSDL, support tools could automatically regenerate the tests and logic and check for compliance with the 1149.1 standard.

If a project, wishing to use boundary scan tests, starts with BSDL, compliance to the standard can be checked automatically. Some initial 1149.1 implementations were incorrect and prevented using boundary scan test tools. A tool monitoring compliance can uncover missing data, or erroneous data, such as including a TAP pin in the boundary register. As always, the earlier a problem is detected, the better.

A natural next step is synthesis. With exception of highly specialized features related to user-defined instructions, boundary scan logic is quite regular and its design is automated readily. This is true even when the functional logic is not synthesized. When a project does synthesize the core logic, typically they desire to automate the entire process and do not wish to manually generate test logic outside their design flow.

Before or during synthesis, test generation can begin. Board tests can be created and optimized long before ASIC fabrication. ASIC tests of boundary scan functions also can be auto-generated. Functional and structural tests can be simplified by insertion of internal scan. This is especially important when boards contain a mix of scan and non-scan devices, because testability problems can occur due to lack of controllability or observability. Make note that board test generation requires some description of the board topology which BSDL does NOT address. Logically, this also should be standardized. An effort to do so, called HSDL (Hierarchical Scan Description Language), is underway. HSDL will be discussed as part of BSDL status.

Boundary Scan Characteristics.

What features of IEEE 1149.1-1990 must BSDL include? As stated, it only specifies variable parameters of the basic elements.

Table 3.4.1 is intended as a representative sample of elements in BSDL, not a complete listing. Basically, the language treats the core logic as a black box with terminal connections to the test logic. BSDL's job is to define the test elements and their connection with no knowledge of the device's internal functions. Further, BSDL does not specify electrical parameters in any fashion. It covers only those factors concerning the logical behavior of the boundary scan standard.

Table 3.4.1. Boundary Scan Characteristics and Their Correlation to BSDL.

Boundary Scan Element	BSDL feature
TAP controller state diagram	This is inherent to the standard, and is <i>not</i> in BSDL.
Bypass register	This is fixed and may not vary. It is <i>not</i> in BSDL.
Addition of optional TRST* pin	Status of TRST* pin incorporation <i>is</i> included.
Device-identification register	BSDL <i>indicates</i> if this optional register is present and <i>specifies</i> its capture value. However, its length is fixed and is <i>not</i> in BSDL.
Physical location of TAP pins	This variable <i>is</i> in BSDL.
Boundary Register	BSDL <i>defines</i> its length, structure, capture data and safe values for output control bits.
Instructions Register	BSDL <i>defines</i> its length and capture value.
Instruction codes	These variables <i>are</i> in BSDL.
Provision of SAMPLE/ PRELOAD, BYPASS and EXTEST instructions	Fixed features of standard, <i>not</i> in BSDL.
Operation of recommended, but optional, and user-defined instructions	Although variable, such operation is <i>beyond the scope</i> of BSDL.
User-defined data registers	BSDL <i>identifies</i> user-defined data registers and <i>specifies</i> their length and associated instructions .

Language Elements.

BSDL, like its VHDL-parent's syntax, is not case-sensitive and allows free-format statements, spanning multiple lines and terminated with appropriate characters or reserved words. Comments contain anything between a "--" symbol and the end-of-line. Thus, for comments, end-of-line is the terminating character. Many boundary-scan characteristics are expressed in VHDL quoted character strings. Each string is linked to a VHDL attribute and must conform to the string's corresponding BSDL syntax. VHDL ignores BSDL-specific syntax; only applications which read BSDL (compiled or not) check for its requirements. BSDL tools fall in two categories. In one, BSDL information is accessed from the compiled VHDL design "data base". By referencing boundary scan attributes, VHDL-based tools can extract desired data. In a device design environment, which includes synthesis and simulation, it would be most convenient to use this all-VHDL flow. The second category, of course, falls in the non-VHDL environment. These programs parse the VHDL looking for BSDL syntax and ignoring all others. It

would be difficult and wasteful here to attempt to act on the full range of VHDL constructs. That is why both the syntax and practices are constrained, making BSDL a subset and standard practice of VHDL. (When many methods exist to model a function, only one is permitted by the "standard practice".) As a result, the front-end (parser) of every process is comparable and simplified, leaving more time to develop and run the main process.

BSDL utilizes three structural units of VHDL; they are Entity, Package, and Package Body. An entity houses device descriptions for VHDL. Part of that component definition is the Boundary-Scan parameters added by BSDL. To prevent duplication of effort, promote consistency and reduce errors, all 1149.1 definitions are provided by a pre-written, standard VHDL package and package body. The package contents are tied to the 1149.1 standard, and should be as constant as the standard itself. This source should be protected from change, accidental or otherwise. Each release of a new standard would necessitate the creation of a new package conforming to it.

An additional package and package body pair may be developed by the user to relay design-specific information. Because of design constraints, added test functions, silicon technology or other reasons, a user may create a library of unique cell definitions or special test structures. By isolating this data in a separate package body, it may be updated without recompiling all the entities which reference the package.

BSDL data contained in VHDL strings has unique syntax, separate from VHDL, and requires its own rules. These rules are specified in simple Backus-Naur Form (BNF) or for more obvious cases, only an example is used. Because the strings can contain many pieces of information, they may become quite long. Therefore, the concatenation operator, '&', is recommended to split the strings into lengths, which easily can be edited, viewed, printed and saved in standard (<133 characters / line) text files.

Status.

BSDL has been proposed by Hewlett Packard (HP) as the standard description language for boundary-scan devices complying with IEEE Standard 1149.1-1990. HP has been working closely with industry to refine and distribute the standard even before submitting it for review by the 1149.1 working group to be included in 1149.1b. Through a combination of need in industry and promotion by HP, BSDL has become a defacto standard, and is supported by many ATE, CAE and semiconductor vendors and customers. The goal for BSDL is that, as new applications need more features, changes will be minor and evolution will be upwardly compatible.

Hierarchical Scan Description Language (HSDL) is currently being proposed by Texas Instruments. HSDL supports all aspects of BSDL, but adds features to describe the interconnection of boundary scannable devices at the board, system or multi-chip module levels and to support interactive debug at the device level. HSDL provides the ability to describe boards, name subsets or supersets of test registers, create symbol tables for test registers or files that use symbolic, named values, prevent illegal states from being established, and so forth. HSDL includes features such as the ability to describe different status values captured by a test register and designate them as "pass" or "fail" values. For interactive use, HSDL allows adding descriptive text to each item in the entity.

HSDL is a strict superset of BSDL. All statements unique to HSDL device entities are optional. Thus, BSDL is accepted by an HSDL translator, and HSDL models can be trimmed down to pure BSDL easily. If HSDL is fed to the BSDL translator, the translator will identify all non-BSDL statements. Those statements then can be deleted without creating syntax errors and without losing or altering the meaning.

3.4.2. Serial Vector Format (SVF).

The Serial Vector Format (SVF) consists of test patterns described as ASCII commands which represent the stimulus, expected response, and mask data for 1149.1 based tests. The need for SVF arose from the desire to have vendor independent 1149.1 test patterns which could be reused in a number of environments, spanning from design verification through field diagnostics. Migrating test vectors from one test execution platform to another, and applying them in multiple phases of test, is difficult or impossible because of all the parsers, translators, and formatters required to process the various internal test vector formats of tools. SVF will allow seamless transfer and use of IEEE 1149.1 vectors. SVF is vendor independent yet flexible enough to adapt to a wide range of applications.

Transition Based Behavior.

The execution of tests via IEEE 1149.1 is controlled by the sequencing of TAP signals on the pins of the devices. Each device's behavior is determined solely by the states of its TAP pins. Boundary-scan tools must maintain knowledge of the sequences required to exert certain behaviors within a device, and where that device is located down the serial scan path.

SVF controls the 1149.1 test bus via commands which transition the TAP from one steady state to another. Rather than describe the explicit state of the IEEE 1149.1 bus on every TCK cycle, SVF describes it in terms of transactions conducted between stable states. For instance, the process of scanning in an instruction is described merely in terms of the data involved and the desired stable state to enter after the scan has been conducted. The capture, update, pause, etc. states are inferred rather than explicitly represented. The data to be scanned in, expected data out, and compare mask are all grouped in an easily understandable manner. A command is provided to support deterministic navigation of TAP states where required.

In addition to supporting higher level depictions of scan operations, Serial Vector Format also supports combined serial and parallel operations. This will allow SVF to accommodate ATE environments where some stimulus response is handled via parallel I/O, and serial signals are accessed via an IEEE 1149.1 control environment.

SVF also supports the concept of scan offsets. Offsets allow a test to be applied to a component or cluster of logic embedded in the middle of a scan chain. In Figure 3.4.2, a device exists in multiple instances on a board. Serially applied tests were generated by the designer which are available in SVF format. To reutilize this test, it is necessary to put all other devices on the scan chain into the bypass mode. The 1149.1 test controller must therefore comprehend 24 instruction register bits before and behind the target

device. Likewise, once in bypass, the devices will introduce 3 data register bits before and after the target device.

SVF allows a header and trailer to be defined once which maintains the instruction register and data registers of the non-targeted devices in the desired bypass state. No modifications are required to the SVF for the device. In Figure 3.4.2, if the same test were targeted towards another device downstream, this would be accommodated merely by changing the headers and trailers.

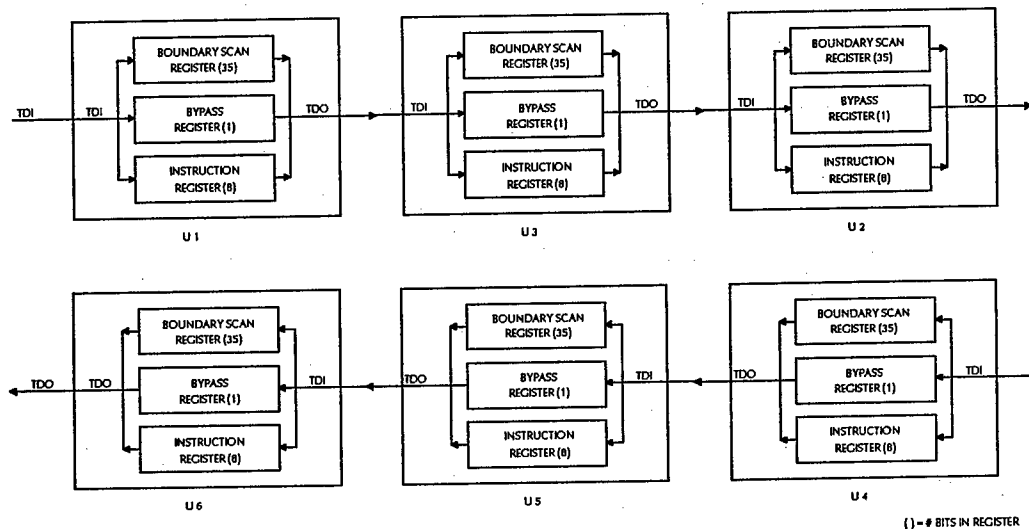


Figure 3.4.2. Scan Chain of Six ICs

The offset approach is capable of installing any instruction and maintaining the data registers in a known state, provided these values are constant for the entire process of applying the SVF device sequence.

SVF Formats. The SVF file is the media for exchanging descriptions of high level 1149.1 bus operations. In general, 1149.1 bus operations consist of scan operations and movements between different stable states on the 1149.1 state diagram. SVF does not explicitly describe the state of the 1149.1 bus at every Test Clock.

The SVF file is defined as an ASCII file which consists of a set of SVF statements. Statements are terminated by a semicolon (;) and may continue for more than one line. The maximum number of ASCII characters per line is 256. SVF is not case sensitive, and comments can be inserted into a SVF file after an exclamation point (!) or a pair of slashes (/).

Each statement consists of a command and parameters associated with that specific command. Commands can be grouped into three types; state commands, offset commands, and parallel commands.

State Commands. State commands are used to specify how the test sequences will traverse the JTAG state machine. The following state commands are supported:

SDR	Scan data register
SIR	Scan instruction register
ENDDR	Define end state of DR scan
ENDIR	Define end state of IR scan
RUNTEST	Enter Run-Test/Idle state
STATE	Goto specified stable state
TRST	Drive the TRST line to the designated level

SDR performs an IEEE 1149.1 data register scan. SIR performs an IEEE 1149.1 instruction register scan. ENDDR and ENDIR establish a default state for the bus following any data register scan or instruction register scan, respectively. RUNTEST invokes a Run-Test/Idle state for a specific number of TCKs. For each of the above commands, a default path through the state machine is used. Each of these commands also terminates in a stable non-scan state.

STATE places the bus in a designated IEEE 1149.1 stable state. A designated path through the state machine can be specified to be followed. TRST activates or deactivates the optional test reset signal of the IEEE 1149.1 bus.

Offset Commands. Offset commands allow a series of SVF commands to be targeted towards a contiguous series of points in the scanpath. Examples would be a sequence for executing self test on a device, or a cluster test where all devices involved in the cluster test are grouped together. The following offset commands are supported:

HDR	Header data for data bits
HIR	Header data for instruction bits
TDR	Trailer data for data bits
TIR	Trailer data for instruction bits

HDR specifies a particular pattern of data bits to be padded into the front end of every data scan. HIR specifies the same for the front of every instruction register scan. Likewise, TDR and TIR specify data to be injected on the back end of each scan. These patterns need only be specified once and are included on each scan, unless changed by a subsequent HDR, HIR, TDR or TIR command.

Parallel Commands. Parallel commands are used to map and apply parallel commands.

PIO	Specifies a parallel test pattern
PIOMAP	Designates the mapping of bits in the PIO command to logical pin names

Parallel commands allow SVF to combine serial and parallel sequences. PIOMAP commands are used by parallel I/O controllers to map data bits in the command into

parallel I/O channels using the ASCII logical pin name as a reference. The PIO command specifies the execution of a parallel pattern application / sample. SVF does not specify any other properties of parallel I/O such as drive, levels, or skew.

Default State Transitions. Table 3.4.2 identifies the default path taken when transitioning from the current state to a specified new state. The defaults may be overridden by specifying an explicit valid sequence using the STATE command.

Example.

An example of a SVF file is shown below.

!Begin Test Program	
TRST OFF;	!Disable Test Reset line
STATE RESET;	!Initialized UUT
ENDIR IDLE;	!End IR scans in IDLE
ENDDR DRPAUSE;	!End DR scans in DRPAUSE
	!
HIR 24 TDI (FFFFFF);	!24 bit IR header
HDR 3 TDI (7) TDO (7) MASK (0);	! 3 bit DR header
TIR 16 TDI (FFFF);	!16 bit IR trailer
TDR 2 TDI (3);	! 2 bit DR trailer
	!
SIR 8 TDI (41);	! 8 bit IR scan
SDR 32 TDI (ABCD1234) TDO (11112222);	!32 bit DR scan
RUNTEST 95 TCK ENDSTATE IRPAUSE;	!RUNBIST for 95 TCK Clocks
SIR 8 TDI (00) TDO (21);	! 8 bit IR scan and check
	! for status bit
STATE RESET;	!Enter Test-Logic-Reset
!End Test Program	

The test begins by de-asserting TRST. The IDLE state is established as the default end state for instruction scans, and DRPAUSE for data scans. Twenty four bits of header and sixteen bits of trailer data are specified for instruction register scans. No status bits are checked. Three bits of header data and two bits of trailer data are specified for data register scans.

Table 3.4.2. Stable State Paths

Current State	New State	State Path
RESET	RESET	RESET
RESET	IDLE	RESET-IDLE
RESET	DRPAUSE	RESET-IDLE-DRSELECT-DRCAPTURE-DREXIT1-DRPAUSE
RESET	IRPAUSE	RESET-IDLE-IRSELECT-IRCAPTURE-IREXIT1-IRPAUSE
IDLE	RESET	IDLE-DRSELECT-IRSELECT-RESET
IDLE	IDLE	IDLE
IDLE	DRPAUSE	IDLE-DRSELECT-DRCAPTURE-DREXIT1-DRPAUSE
IDLE	IRPAUSE	IDLE-DRSELECT-IRSELECT-IRCAPTURE-IREXIT1-IRPAUSE
DRPAUSE	RESET	DRPAUSE-DREXIT2-DRUPDATE-DRSELECT-IRSELECT-RESET
DRPAUSE	IDLE	DRPAUSE-DREXIT2-DRUPDATE-IDLE
DRPAUSE	DRPAUSE	DRPAUSE-DREXIT2-DRUPDATE-DRSELECT-DRCAPTURE-DREXIT1-DRPAUSE
DRPAUSE	IRPAUSE	DRPAUSE-DREXIT2-DRUPDATE-DRSELECT-IRSELECT-IRCAPTURE-IREXIT1-IRPAUSE
IRPAUSE	RESET	IRPAUSE-IREXIT2-IRUPDATE-DRSELECT-IRSELECT-RESET
IRPAUSE	IDLE	IRPAUSE-IREXIT2-IRUPDATE-IDLE
IRPAUSE	DRPAUSE	IRPAUSE-IREXIT2-IRUPDATE-DRSELECT-DRCAPTURE-DREXIT1-DRPAUSE
IRPAUSE	IRPAUSE	IRPAUSE-IREXIT2-IRUPDATE-DRSELECT-IRSELECT-IRCAPTURE-IREXIT1-IRPAUSE

In the example above, a single device in the middle of the scan path is targeted. Notice that the targeted device has 3 devices before it in the scan from the 24-bit IR header (3 x 8-bit IR) and the 3 bit DR header (3 x 1-bit DR). The targeted device also has 2 devices after it in the scan path from the 16-bit IR header (2 x 8-bit IR) and the 2 bit DR header (2 x 1-bit DR). After the header and trailer offsets are established all subsequent scans are the concatenation of the HEADER, scan data, and TRAILER bits. The targeted device supports Self Test which is initialized by scanning a hex 41 into the instruction register, followed by a hex ABCD1234 into the selected data register. The targeted device BIST is then executed by entering the Run-Test/Idle state for 95 clocks. Next, a new instruction is scanned in and the status bits compared against a deterministic value to determine pass/fail.

4. TEST BUS APPLICATIONS

4.1. Test Bus Architecture And Interfaces.

Test Bus architecture and interface designs must consider the use and types of buses necessary for support of production, mission, depot, and maintenance applications. The following subparagraphs describe efforts undertaken by industry and government programs in support of this concern.

4.1.1. Avionics System Architecture.

Advanced avionic system platforms are pursuing the use of common buses, modules, interfaces, and microprocessors to meet application, throughput, weight, reconfiguration, fault tolerance, fault detection and isolation requirements of the next decade. Work has been completed by the Joint Integrated Avionics Working Group (JIAWG) under the direction of the Tri-Services to define requirements for a Common Avionics Baseline (CAB). This baseline established requirements for common modules, buses, test strategies, and the storage of fault data within module memory.

System common buses provide for standardization of bus interface connection, control, data exchange, and bus testing. The JIAWG has identified several buses for use in advanced avionics architectures. These buses include:

- MIL-STD-1553
- High Speed Data Bus
- High Bandwidth Interface

System Buses.

The MIL-STD-1553 data bus provides control and data communications between processing resources and off-the-shelf equipment, and between avionic sensors and subsystem elements. The MIL-STD-1553 data bus is a serial, command/response, multiple terminal data bus capable of providing a data transfer rate of 1 million bits per second. Redundant 1553 data buses are typically employed within system architectures to provide for tolerance against battle damage.

The High Speed Data Bus provides a high speed message passing capability between system elements. The High Speed Data Bus is a fiber optic, linear, token passing data bus capable of a 50 million bits per second transfer rate. Redundant buses are also used to provided for tolerance against battle damage.

The High Bandwidth Interface provides a high-speed fiber optic link for the communication of data between enclosures within the system; enclosures being the grouping of subsystems to form an integrated signal and data processing element.

Figure 4.1.1 illustrates an advanced avionics architecture which makes use of this system bus concept.

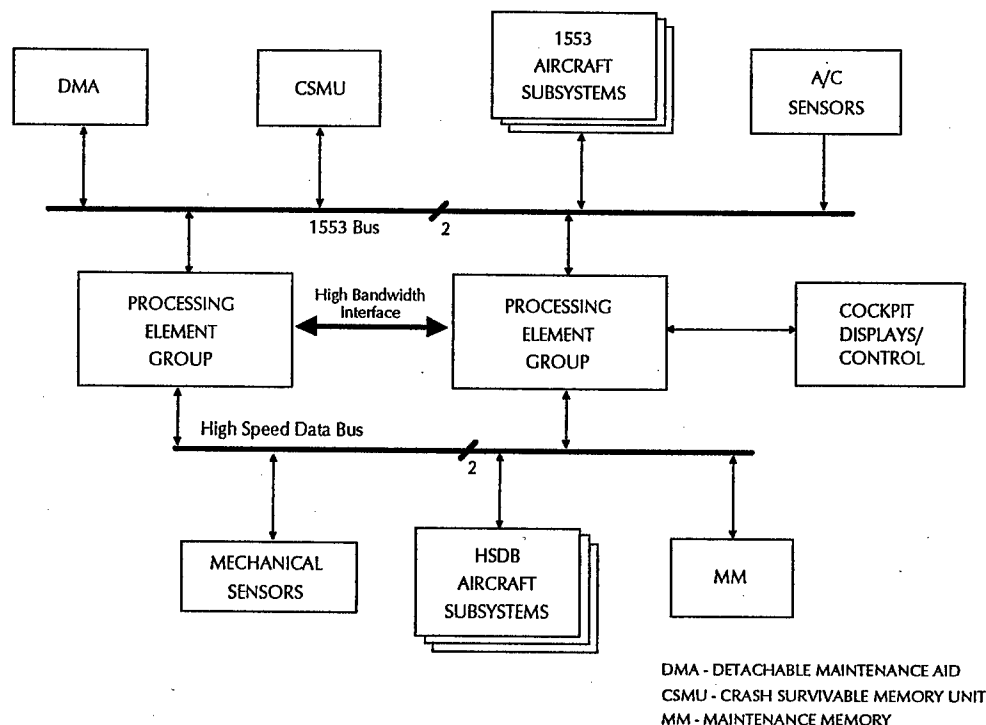


Figure 4.1.1. An Advanced Avionics Architecture

System diagnostic applications utilize the HSDB and 1553 system buses to pass control data to and receive status information from avionic subsystems. Control data passed to subsystems can initiate subsystem internal built-in self-test diagnostic applications, thus allowing a subsystem to determine its own operating status. Via the system buses, this status can then be requested for transmission to system fault management applications for subsystem status assessment, allowing for reconfiguration of system resources around faulty subsystems. System failure information can be transferred, via the system buses, to on-board maintenance memory to preserve environment parameters during the time the failure occurred. The bandwidth of each system bus currently supports system throughput requirements, and with enough reserves for system expansion.

4.1.2. System to Module.

Meeting system capability requires an architecture dominated by processor and digital logic. The use of common modules and built-in test strategies are demanded. Common module sets and data buses are being defined and developed to provide the processing and throughput requirements needed to meet future mission needs. By mapping subsystem applications on common modules and connecting these modules via common module data buses to form processing groups, it has been found that logistics and maintenance can be reduced, while at the same time, system fault tolerance and testing capability can be increased. Needless-to-say, the use of common modules reduce the types of modules required and thus reduces the number of modules needed to support subsystem applications.

The JIAWG has defined several common module architectures which support future system required capabilities. These common modules fall into seven categories: 1) 16 bit/32 bit common avionics processors, 2) user console interface, 3) power supplies, 4) memories, 5) graphics and signal processors, 6) data network modules, and 7) data bus interface modules. The JIAWG has also defined requirements for buses which interface to common modules. These buses consist of:

- Parallel Intermodule (PI) bus
- Test and Maintenance (TM) bus
- Local Memory bus
- Signal Processing (SP) bus

The Avionics Commonality Working Group, an extension of the JIAWG, continued previous work performed by the JIAWG on the TM bus by identifying extensions to the bus. The Society of Automotive Engineers (SAE) has established a working group to define electrical requirements for a Module Test and Maintenance (MTM) bus and avionics extensions. Final TM bus requirements are expected to be formed from the work of both of these working groups.

Module Buses.

The Parallel Intermodule bus provides a message and data communications path between modules supporting a system subsystem application. It is a 16-bit parallel, redundant backplane bus which operates at a 12.5-MHz clock rate. The PI bus provides a backup capability to the Test and Maintenance (TM) bus for control and status information in the event of TM bus failure. The number of modules supported by the PI bus is currently 32 modules.

The Test and Maintenance bus provides the primary test and maintenance path for modules supporting a subsystem application. It is a serial, linear, master/slave protocol bus for test and maintenance control, data, and status and operates in a multi-drop environment. Clock rate is 6.25 MHz and redundant buses are generally used within a system's architecture. The number of modules supported by the TM bus architecture is currently 61 modules.

The Local Memory bus provides a memory expansion capability to a common avionics processor when on-board processor memory is not capable of supporting subsystem application requirements.

The Signal Processing bus provides for high-speed, data intensive, low-latency transmission between signal processing modules supporting subsystem application requirements.

Figure 4.1.2-1 illustrates the connectivity of these data buses within a subsystem environment.

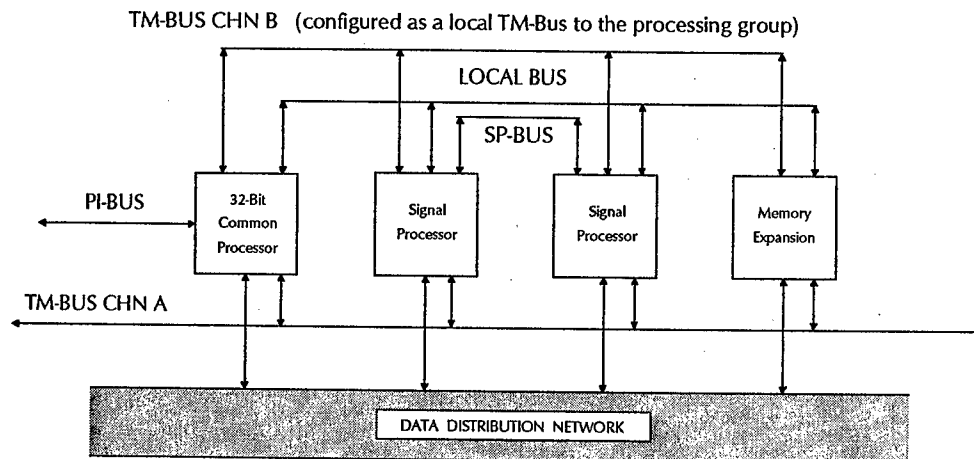


Figure 4.1.2-1. Subsystem Bus Architecture

Subsystem buses, like system buses, provide a medium for transmission of control and status information between modules, which make up a subsystem, and between subsystems. Via the subsystem buses, individual module self-test applications can be initiated and the resulting status from testing obtained for status assessment. Test results can then be transferred to a module embedded fault log to preserve environment parameters at the time the fault occurred.

Module Fault Log.

The JIAWG has established the format and data content for common module embedded fault logs. Extensions to this format and data content have been defined which provide additional information to assist vehicle and maintenance decisions. Information contained within an extended module fault log includes (*; extensions to the JIAWG):

- Name plate information
 1. part, serial number
 2. configuration, stock number
 3. accumulated runtime, power on/off cycles
 4. slot, rack, bay, A/C ID, side
 5. log fault pointer
 - *6. log rollover indicator, fault log count
- *• Maintenance history information
 1. repair count
 2. first and second highest replaced part
- Module fault information for each fault occurred
 1. fault code, bit mode
 2. year/month/day/hour/minute of failure

3. temperature and environment data
 - *4. bus status information
 - *5. module operating states
 - *6. subsystem reporting failure and task executing at time of failure
 - *7. module mission capability
 - *8. detailed test parameters of test detecting the failure
- *• Fault log checksum information

Figure 4.1.2-2 illustrates a module (LRM) architecture which uses the TM bus and PI bus to transmit control data to and receive status information from module embedded built-in testing applications. Note; through subsystem bus interfaces located on a module, IEEE 1149.1 test control and data can be issued to trigger embedded integrated circuit (IC) testing logic and collect testing results.

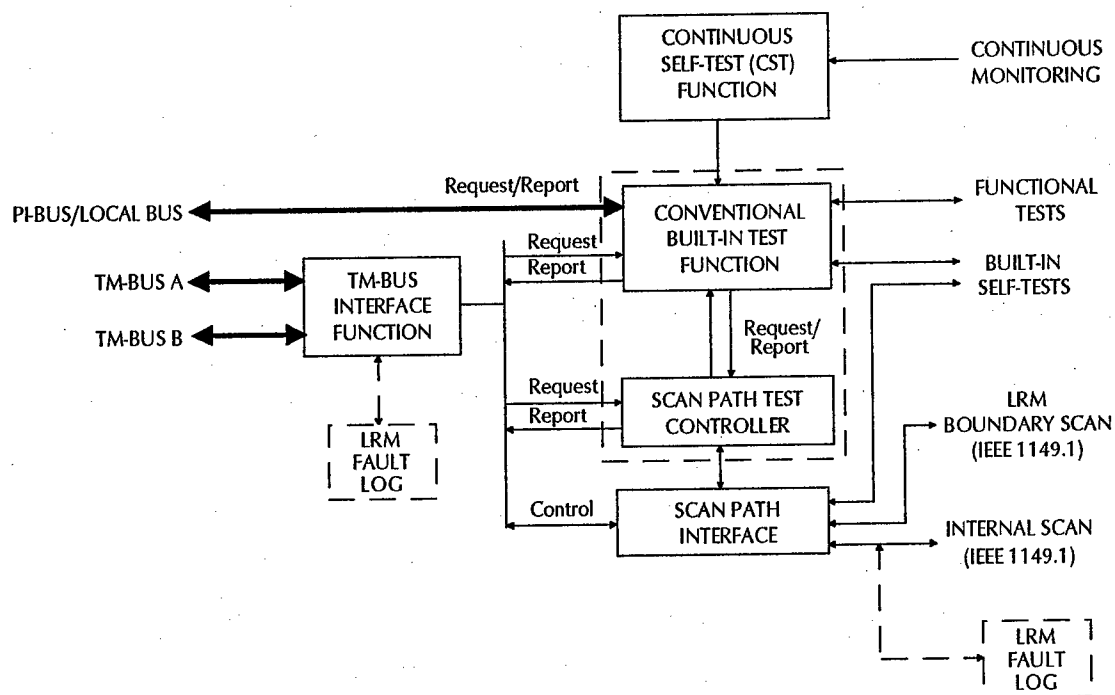


Figure 4.1.2-2. Module Test Bus Architecture

Module Testing.

The types of module self-test applications which can be initiated via the subsystem buses fall into the following categories:

- Start-up Built-in Test (SBIT)
- Periodic Built-in Test (PBIT)
- Initiated Built-in Test (IBIT)
- Continuous Self-Test Built-in Test (CBIT)

SBIT. Executes automatically upon power being applied to the module or upon module reset. SBIT testing applications usually support module warm start and cold start execution.

PBIT. Executes periodically at some scheduled frequency and is generally non-intrusive in its execution. It is typically capable of being interruptible by higher system applications

IBIT. Executes upon being initiated by system applications. Test initiation is generally performed over the TM bus, however, backup test initiation can be provided over the PI bus, local bus, or SP bus.

CBIT. Executes continuously as long as power is applied to the module. It consists of hardware monitoring functions and test software executing on the module during processor idle time.

Fault Data Impacts on Module Buses.

The status information received over the subsystem buses from the execution of built-in tests, provide information to subsystems and the system which allows them to perform fault verification, fault filtering, fault correlation, reconfiguration, and prognostic applications to assist in meeting mission objectives. The types of information needed by these applications and received via subsystem buses include:

- Module bus status (PI, TM, Local)
- Module health (fully capable, mission capable, degraded, or inoperable)
- Module BIT status and BIT mode being executed
- Operating state of the module (on-line/off-line, enabled/disabled)
- Module type (processor, memory, interface, etc.)
- BIST register of module integrated circuits (if applicable)

The impact this information has on the design of bus architectures used at the subsystem level has been studied and the conclusions are herein presented.

Advanced avionic platforms require a large number of common modules to support all vehicle subsystems. Limitations in TM bus connections require a bus architecture which is hierarchical in nature. This requirement is driven by bus master/slave protocol, the transfer rate of the bus, the type of information required over the bus, and the frequency at which the information is requested. The TM bus has been targeted as the primary test

bus by the JIAWG and test initiation and test status results are to be issued over the TM bus. Current TM bus mastership arbitration and channel switching schemes (channels A and B) impact bus architecture as the number of common modules on the bus increase. The current number of modules which can be connected to the TM bus is 61, and there can only be one master of the bus at a time. All other modules on the TM bus are slaves to the bus master and must be polled for their status information. To support avionic applications, such as reconfiguration, flight control decision making, etc., fault detection and reporting must be received within time frames which allow flight-critical/safety decisions to be made (generally 3 seconds). The JIAWG has specified the duration at which periodic built-in testing (PBIT) will execute to be 1 second (in 5-ms durations). This time period would allow system applications to react to faulty conditions. The results from a performed study found that as the number of modules connected to the TM bus increased, the longer it took to determine faulty modules connected to the bus due to bus master/slave protocol. It was concluded from the study that a hierarchy of TM buses, with a typical module polling sequence of 1-ms, would be necessary to transmit the required status information (identified above) from modules to subsystems and to system fault management applications to support flight-critical/safety operations in systems which employ a large number of common modules connected to the TM bus.

4.1.3. Module to IC.

Integrated Circuit (IC) manufacturers embedding the IEEE 1149.1 test bus architecture within their designs are providing subsystem and system applications with the capability to support fault detection and fault isolation requirements commanded by current and future system platforms. Work is currently under way by several IC manufacturers to interface the IEEE P1149.5 Test and Maintenance (TM) bus with the IEEE 1149.1 to allow test control and data to be issued from module test applications to module ICs. Via the IEEE 1149.1 test bus, IC BIST can be initiated and the results obtained to determine IC performance measures. Figure 4.1.3 illustrates a design which uses envisioned fault detection and fault isolation capabilities and the IEEE 1149.1 test bus architecture to enhance IC testing. This design makes use of the following concepts:

- Each IC will contain device BIST
- The IEEE 1149.1 Instruction Set Architecture (ISA) will be supported
- Internal scan and boundary scan operations based on the IEEE 1149.1 test bus architecture (with design enhancements) will be supported
- The ability to force BIST failures via the IEEE 1149.1 test bus will exist
- BIST performance will be placed in a BIST register which is accessible via the IEEE 1149.1 test bus

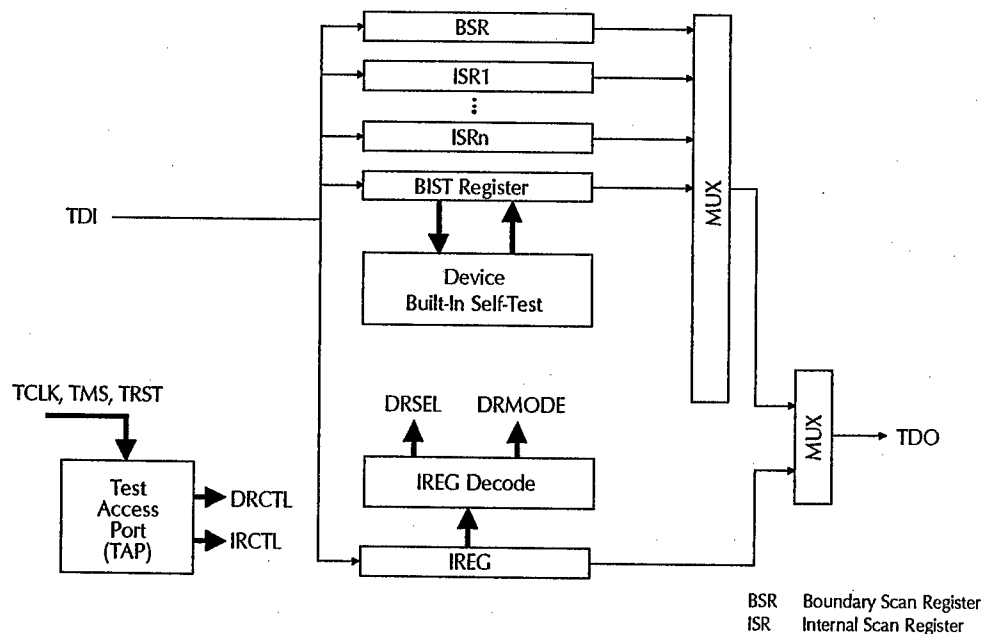


Figure 4.1.3. Integrated Circuit Test Bus Architecture

It is envisioned that each IC will be designed with hardware and software features which support testing via the IEEE 1149.1 test bus in any test environment (factory, field, depot) and at any level of equipment integration (system, subsystem, module). ICs should support the standard IEEE 1149.1 test functions and extensions to the standard. Standard test functions include a TAP for scan protocol interpretation, instruction and data registers for control and execution of scan operations. Extensions to the standard includes the ability to initiate BIST and collect the results in a BIST register, generate random test patterns, and perform signature analysis applications.

4.2. Specific Application Examples.

The previous section introduced system test bus and interface architectures. Specific applications employing these architectures are discussed in the following subsections and include:

- F-16 Modular Mission Computer
- F-22 Vehicle Management System Test Bus Architecture
- F-22 Radar Array Power Supply
- Aladdin Test Bus Architecture
- Solid State Recorder (SSR)

4.2.1. F-16 Modular Mission Computer.

Hardware Description.

The F-16 Modular Mission Computer (MMC) is an integrated mission computer which provides advanced computing for the F-16's avionics and weapon systems. The MMC performs computing for weapons delivery, navigation, Head-Up Display, and helmet mounted display. It replaces three computers in the current configuration at less weight and provides additional computing power and space for future growth. A MMC consists of a data processing, avionics display, aircraft specific I/O, and power supply modules. MMC modules communicate on the backplane using a common set of backplane interfaces. Processing modules communicate over a dual redundant Parallel Interface Bus (PI-Bus) and a dual redundant Test and Maintenance Bus (TM-Bus). Real-time module debug capabilities are embedded in the processing modules and interface with the Computer Development System via IEEE-488 bus. The MMC architecture is shown in Figure 4.2.1-1.

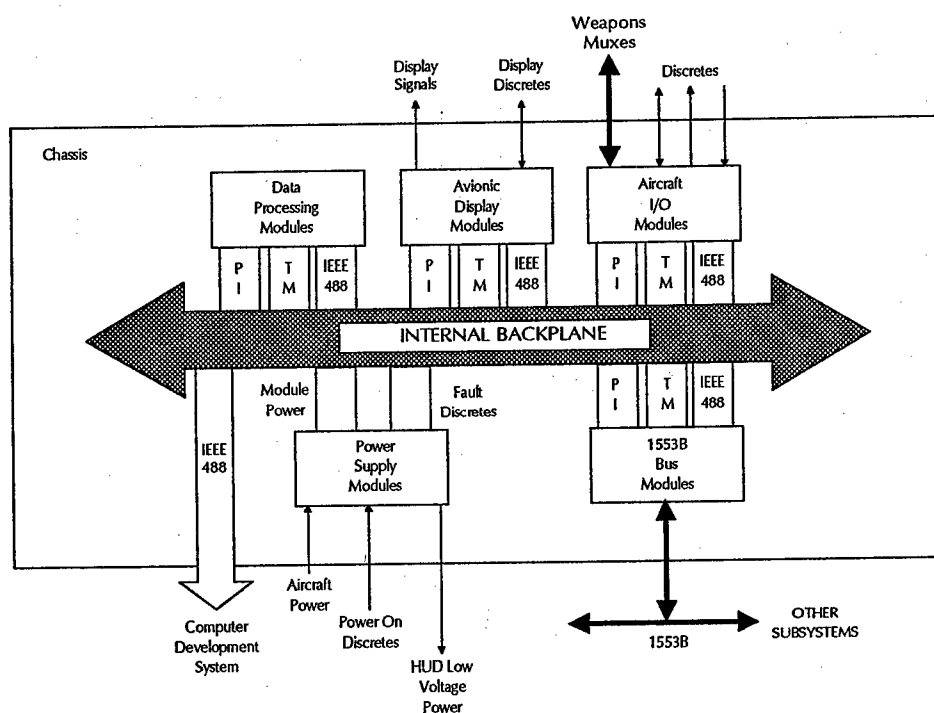


Figure 4.2.1-1. F-16 MMC Architecture

The MMC test architecture is hierarchical in nature starting from the system level down below the IC level. At the system level, system and module health is collected and processed for logging and reporting. At the system level, test commands and status information is transmitted and received via the MIL-STD-1553B system buses. System level command and status information is communicated on the 1553B bus via 1553B data transfer messages.

Module health and status information is collected on each module and communicated via the TM-Bus. Processor modules contain a TM-Bus interface function which communicates module test information to the system TM-Bus master. The TM-Bus interface can be configured to allow the module to communicate as a TM-Bus master or TM-Bus slave on the MMC backplane. The TM-Bus interface can control and communicate various information including, initiating module BIT, module test status, module fault log information, and even functional data. The TM-Bus interface can also serve as an interface to communicate with the module internal IEEE 1149.1 test bus. Autonomous Built-In Test (BIT) operates in several modes including Start-up BIT (SBIT), Periodic BIT (PBIT) including continuous hardware monitors, and Initiated BIT (IBIT). SBIT includes extensive hardware Built-In Self Test (BIST) capabilities for the ASICs and module. All ASIC and module BIT information is collected, filtered and logged in a hierarchical results table which is available for system interrogation. The MMC test bus architecture is shown in Figure 4.2.1-2.

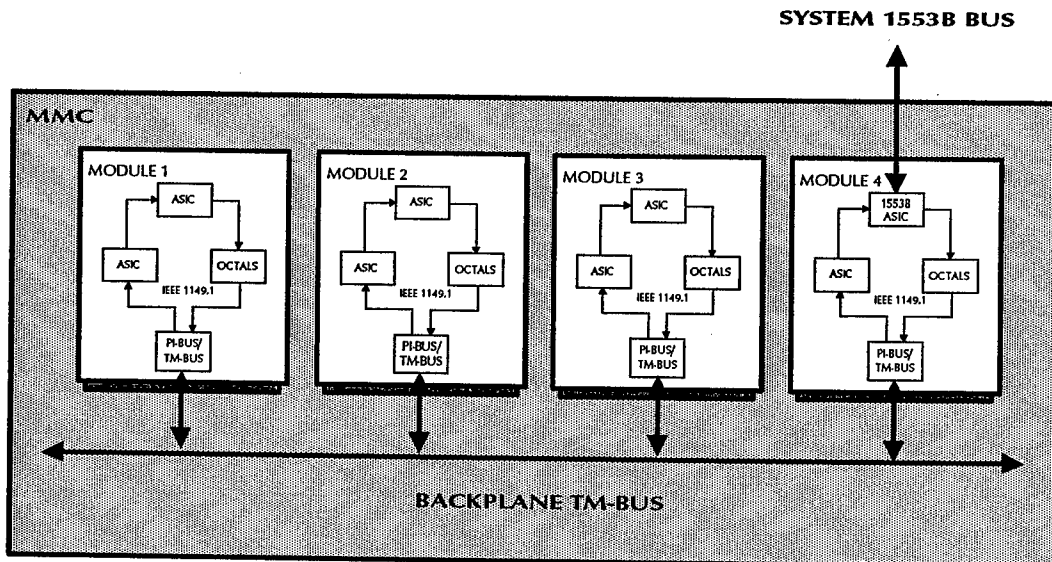


Figure 4.2.1-2. F-16 MMC Test Bus Architecture

Each MMC module has a significant amount of embedded test logic. Each ASIC has a superset of the IEEE 1149.1 test bus and boundary scan implemented along with internal scan and BIST logic. The PI-Bus/TM-Bus Unit (PTU) ASIC consolidates many of the module level test features and serves as both the external test interface via the TM-Bus and the module BIST controller. The MMC modules are one of the first modules to utilize the IEEE 1149.1 for module BIST. A module BIST controller function contained in the PTU and 1149.1 test patterns are stored in a reserved portion of the fault log memory. Each ASIC also implements BIST to verify internal core logic.

All ASIC test resources are available via the IEEE 1149.1 test bus. This includes BIST, internal scan, and boundary scan logic. Internal scan was implemented at key points to provide controllability and observability of device registers, internal state logic, and special logic embedded for software/hardware integration. The 1149.1 ICs are all connected in one serial ring on the module and controlled by a 1149.1 master function in the PTU ASIC. To support module design verification, troubleshooting, and manufacturing tests on Automatic Test Equipment (ATE), the 1149.1 test bus can alternatively be controlled by an external master. An external master present pin on the PTU ASIC, enables a multiplexer to drive test data into the PTU (now acting as a slave on the 1149.1 bus) and through the other 1149.1 devices. A Data Processor 32 (DP32) module is shown in Figure 4.2.1-3.

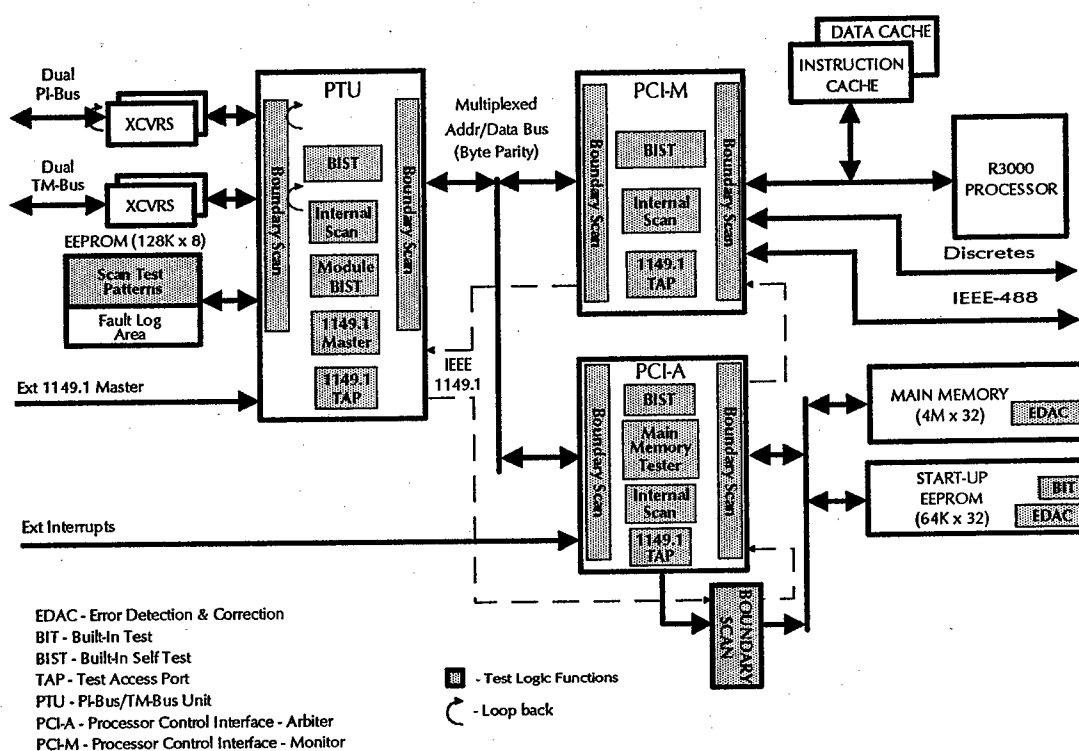


Figure 4.2.1-3. F-16 MMC DP32 Module Test Architecture

Summary.

As discussed above, a significant amount of MMC resources are dedicated for test. This includes, BIST and IEEE 1149.1 circuitry in ASICs, module BIST and test logic, TM-Bus interface logic, fault log memory, BIT code in startup ROM, test signals on the backplane, and so on. All these resources are implemented to achieve the high levels of module level fault detection and fault isolation required to support two level maintenance and reduce life cycle costs.

The F-16 test bus architecture is a consistent hierarchy from IC to module to system. Starting at IC level, 1149.1 is used between ICs on a module, the TM-Bus interconnects

modules on the backplane, and 1553B is used to communicate with other subsystems. This architecture was developed by the equipment prime contractor to ensure consistent test access to all levels. Specific test requirements were integrated and allocated down to the system, module and IC levels. An important side effect of this approach is that all module and IC types will have the same set of test features, no matter how experienced the designer is in implementing testability. Consistent requirements and implementation across module types serves to minimize special test equipment by providing a common test interface both to on-aircraft and off-aircraft test environments. Savings are also realized by reduced automatic test equipment costs and labor costs because UUTs are more self testable.

4.2.2. F-22 Vehicle Management System Test Bus Architecture.

The Vehicle Management System (VMS) provides general purpose data processing and I/O control for F-22 aircraft. The VMS subsystem consists of many different module types from various vendors. Modules communicate on the VMS backplane via common interfaces such as the PI-Bus as well as aircraft specific communication buses and discrete I/O signals. The VMS subsystem is unique from other Joint Integrated Avionics Working Group (JIAWG) like architectures in that it does not use a dedicated backplane test bus. Test information is communicated between modules using the PI-Bus, which primarily functions as the functional communications bus. The system architecture is shown in Figure 4.2.2-1.

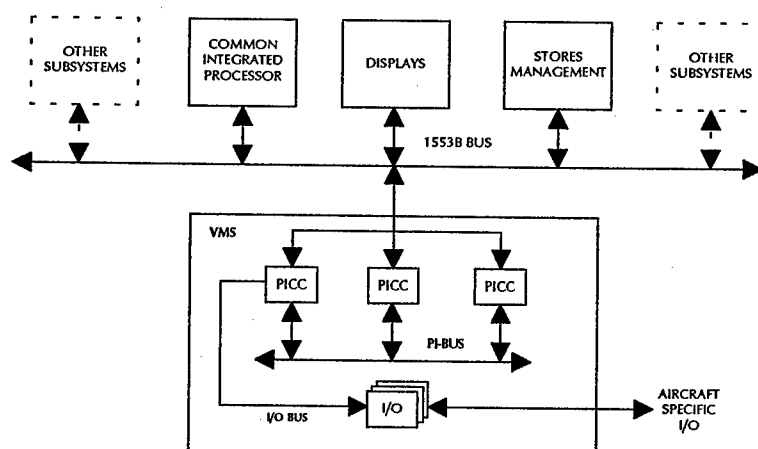


Figure 4.2.2-1. F-22 Vehicle Management System Architecture

This discussion will concentrate on the Processor Interface Control and Communications (PICC) module which is used in multiple locations in the F-22 avionics. The PICC module is based on a MIL-STD-1750A processor and supporting chip set. External module communications are implemented via a PI-Bus, 1553B bus, and an aircraft specific serial I/O bus. Other external interfaces include digital and differential discrete I/O signals. Each of the PICC ASICs implement the IEEE 1149.1 test bus which is routed to the module connector.

The ASICs implement nearly all of the processing, I/O, and glue logic on the PICC module. Boundary scan in these devices provides almost complete control and observation of all the module internal nodes. In addition to the standard 1149.1 features, the ASICs implement 1149.1 extensions to provide additional test and debug capabilities. For example, all ASICs implement the 1149.1 INTEST instruction which allows the IC core logic to be stimulated via boundary scan. Also, most ASICs have internal scan at key points to support IC design verification and fault detection.

The most significant 1149.1 extension is the addition of on-chip emulation logic in the 1750A processor. Emulation logic is accessible via the 1149.1 test bus to control the processor for software integration and hardware testing. Via the 1149.1 test bus, the processor can be commanded to HALT, RUN, SINGLE-STEP, examine/modify registers, and examine/modify memory. This capability is invaluable for test because the processor is not required to execute software to perform these operations. Typical processors must be capable of executing a debug monitor program (which itself is software) to perform these operations. But executing software may be impossible if common hardware faults such as a data bus stuck-at fault exists. Also important is the fact that the emulation actions are transparent to the application program because a debug monitor or memory is not required to control the 1750A processor.

Summary.

System level test information is communicated via 1553B buses between subsystems and separate physical boundaries (boxes) as is typical of most systems. The box-to-box 1553B interfaces are usually redundant so information can always be communicated even in the presence of faults. The decision to not require a module level test bus on the backplane was a subsystem architectural decision made by the prime equipment system engineers based on cost, weight, and power. It was decided that test information would be communicated on the functional backplane bus (in this case the PI-Bus). This approach is acceptable for many systems, and in fact most older systems used functional buses to communicate test information. However, there are several drawbacks to this approach; 1) the test information must be accommodated in the bandwidth of the functional bus (the test information is usually low bandwidth), 2) there is no dedicated test bus to interrogate, log or report faults if the functional bus is inoperative, 3) some module types utilize different functional buses and therefore there is no common bus to communicate test information to/from all module types.

The PICC module contains extensive module test resources including BIST, BIT software, and 1149.1. However, the 1149.1 test bus is only utilized for module integration, factory, and depot test and therefore is not connected on the backplane. There are several reasons why the 1149.1 test bus is not used in the VMS chassis:

1. There is no system-wide 1149.1 master to control the 1149.1 slave modules
2. The VMS system has additional slots for growth and therefore module slots may be empty and result in an open test bus ring
3. The number of VMS modules on a backplane would make a single test bus ring unwieldy and impractical

The PICC module actually has two 1149.1 test bus rings. One contains only the 1750A processor, the other contains the remaining 1149.1 ICs. This partitioning was chosen so that the software integration test system need only control the 1750A processor on the 1149.1 test bus. Control of the other ICs on the 1149.1 test bus was not required for software integration. The 1750A processor 1149.1 ring is routed to a top connector used only for software integration and to the module functional connector. All other ICs are on the second 1149.1 ring which is only routed to the functional connector. During module factory and depot test the two 1149.1 rings on the functional connector are connected in series to form one 1149.1 ring.

4.2.3. F-22 Radar Array Power Supply.

Application Architecture Description.

The F-22 Radar Array Power Supply (APS) provides voltage regulation and EMI filtering for the F-22 Radar system. The APS, shown in Figure 4.2.3, consists of two (2) Input Filter modules, ten (10) Array Power Supply Modules, and an Array Power Supply Control (APSC) module. The APS accepts and conditions +270 VDC Aircraft Power, producing the various DC voltages required for the radar array and for APS internal operation.

The F-22 employs a two-level maintenance concept and requires on-aircraft module level fault isolation via Integrated On-Board Diagnostics (IOBD), with minimal manual procedures and no external test equipment. The APS supports the F-22 IOBD by providing Initiated as well as Periodic and Concurrent Built-In Test functions.

APS BIT consists of IC self-test (BIST), functional or module self-tests, and coordinated Radar System functional testing. Control of the APS BIT tests is typically autonomous during radar operation; but individual tests or blocks of related tests can be executed under external control to support both radar system-level tests and dedicated fault isolation testing on the ground.

Test Bus Implementation.

APS BIT operations are controlled locally by the APSC module. The APS communicates with the rest of the radar system via two unidirectional RS-422 buses (command and status). Primary APS-internal communications are implemented by a four wire serial Test and Maintenance (TM Bus) bus utilizing a subset of the IEEE P1149.5 serial protocol and instruction set. The primary, or "master" control of this TM bus resides on the APSC module, with the slave units on each of the twelve additional modules. Additional discrete wiring between the APSC and other modules of the APS are for high priority status and control signals.

The APSC communicates with both the APS power supply modules and the Input Filter Assembly (IFA) modules via the Test and Maintenance bus. Each module accepts BIT commands and provides module status information on request via the TM bus. The inter-module communications as shown in Figure 4.2.3 include discrete signals such as:

1. SAFETY SHUTDOWN (Radar to APSC)
2. 270 ENABLE #1, 270 ENABLE #2 (APSC to IFA(s))
3. MODULE DISABLE (Drain & Bias) (APSC to Power Modules)

to provide dedicated (and fail-safe) data path control of the modules in case of loss of RS-422A or TM bus serial communications.

In addition, signals to the APSC module include:

1. IFA FAULT #1, IFA FAULT #2 (IFA(s) to APSC)
2. Bias Bus and Drain Bus (sample) (APS Power Bus to APSC)

to provide critical status to the APSC in real-time for immediate reaction to potentially catastrophic fault events, eliminating the latency associated with normal TM bus communications. These signals are also used to assist in APS and Radar Array fault isolation.

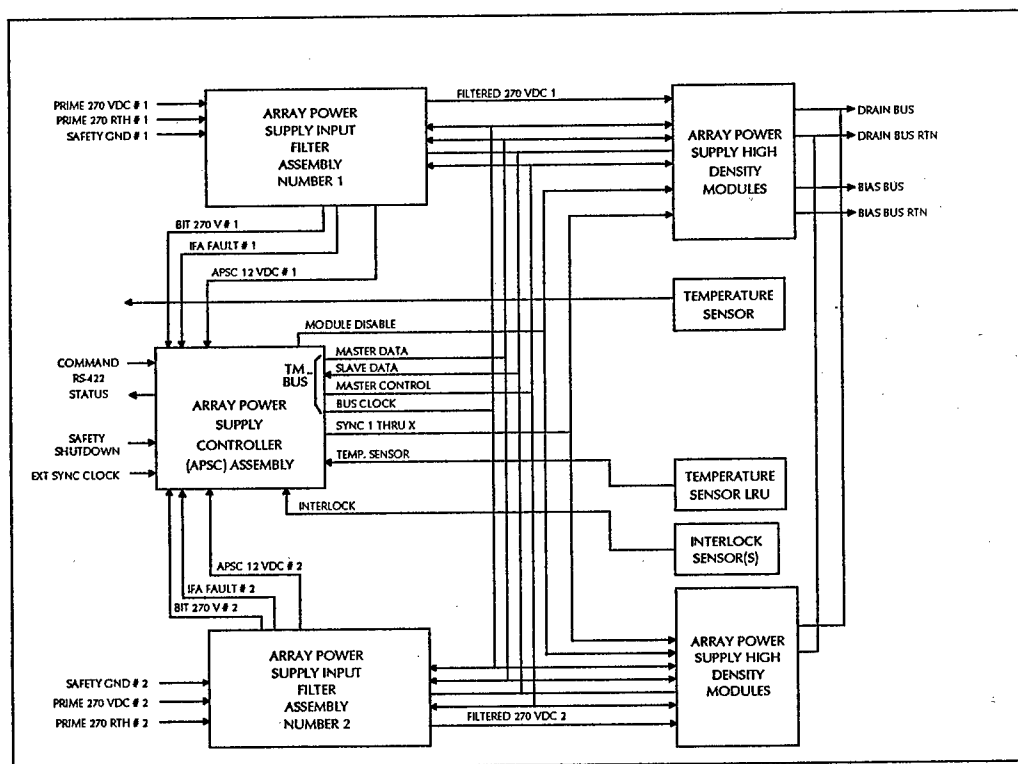


Figure 4.2.3. Array Power Supply Block Diagram

The APSC monitors itself and the associated modules of the APS via combinations of active tests and passive monitoring of signal and hardware status. Most active tests, providing known stimuli and conditions, are executed in dedicated BIT (Start-Up or Initiated BIT). Periodic or Concurrent BIT is executed during APS operation, where

testing is primarily "passive monitoring", non-interfering with operational processes. On request, the APSC communicates with each module via the TM bus, consolidates results, and provides the results to the radar via the RS-422A Status Bus. At the radar (or aircraft) level, the results are integrated with other radar and aircraft information to determine the source of the fault.

Each APS module includes an on-module fault log. Upon aircraft or radar confirmation of a detected fault, and isolated to an APS module, the APSC is instructed to store data regarding the fault, configuration information, and any other pertinent data in the faulty module's error/fault log. The APSC also autonomously records pertinent APS module information in the fault logs. This is information which the radar or aircraft may not require or need not normally see; e.g., over temperature events or total runtime per module.

Less detailed information regarding the fault, usually reformatted as a function of radar operation and modes, is also stored on the aircraft's Data Transfer Cartridge for later retrieval and maintenance actions and/or analysis.

Summary.

The F-22 APS test-related communications are accomplished with a unidirectional RS-422, a modified (subset) IEEE P1149.5 TM bus, and discrete signals. The BIT and diagnostic features of the APS support a two-level maintenance concept with minimal manual operations and external test equipment at the Organizational level of maintenance. On-module fault logging includes temporal data (time, configuration, temperature etc.) to help reduce false alarms and Retest OK's and to support accelerated diagnostic maturation.

In the case of the F-22 APS, the Test and Maintenance bus is not a full implementation of the IEEE P1149.5 TM bus for several reasons. They include the fact that at the time of APS design the standard was still in flux and had not yet been ratified. They also include the fact that the APS operational and diagnostic needs did not require or justify the full implementation of the proposed standard.

The implementation of a subset of the TM bus was accomplished as a cost saving trade-off. It is more likely that the bus implemented would have complied with the IEEE standard in it's entirety if the standard had been ratified prior to APS design. It is also more likely that it would have been used if off-the shelf designs for implementing the interface had been available.

Discrete signals were used to supplement the TM bus to avoid any latency associated with structured bus communications protocol. These signals were used only for critical signals and similar implementations can be anticipated in other systems with critical latency requirements.

Standard buses (i.e. RS-422) are still being used for higher-level test communications. This is a cost/space/weight/power saving measure and similar trade-offs should be anticipated in other systems. The key concern in using such a bus should be the fault tolerance (or lack of it) of the communications mechanism. When fault tolerance is

critical, the use of a dedicated test bus may be justified. This was not so in the case of the F-22 Radar System Array Power Supply.

4.2.4. Aladdin Test Bus Architecture.

The Aladdin parallel processing computer is a very high speed and high density computer for embedded military applications. The primary application of Aladdin is automatic target recognition for intelligent weapons and avionics. It employs silicon-on-silicon packaging, 3-dimensional memory packaging and 100 Mhz operation to achieve 500 MIPS scalar processing and 2 GFLOPS floating point processing performance in a 4.5 inch diameter cylinder which is less than 6 inches high. An Aladdin processing cluster consists of five Basic Processing Modules (BPMs), each providing 100 MIPS/400 MFLOPS performance. BPMs consist of a R4000 RISC scalar processor, a Vector Co-Processor (VCP) ASIC, two Processor Interface Chip (PIC) ASICs, six Crossbar (XBAR) switch ASICs and one megabyte of SRAM. All devices on the BPMs are mounted on a silicon substrate and interconnected via Tape Automated Bonding (TAB).

The scalable modular architecture is achieved by interconnecting the five BPMs over a standard backplane interface. All Aladdin memory is addressable over this bus under a unified address space. BPMs can be added or deleted to satisfy applications requiring more or less throughput. High speed sensor specific ports are provided on each BPM to satisfy very high data rate sensors. Each BPM contains one megabyte of high speed static RAM organized into eight independent banks and connected to the processing resources through a 6x8 crossbar switch. Up to six of the eight memory banks can be accessed simultaneously through the crossbar switch. The processor interfaces to both the crossbar switch and the BPM-to-BPM bus via the Processor Interface Chip (PIC) ASIC. The PIC provides both block transfer and individual word transfers between the RISC processor and memory, within or between BPMs. Vector processing is provided via the Vector Co-Processor (VCP) connected directly to the crossbar switch. The VCP has three memory ports, providing sufficient bandwidth to and from memory to support the operation of two arithmetic-logic units, two multipliers, three address generators, and other on-chip arithmetic resources. Two high speed I/O ports are also implemented so that sensor data and processing results can be written to and read from memory simultaneously with on-board processing. The BPM architecture is shown in Figure 4.2.4-1.

A superset of the IEEE 1149.1 test bus and boundary scan was implemented on each of the ASICs to provide a common test access method for ASIC and BPM testing as well as a method to access internal structures. Internal scan was implemented at key points to provide controllability and observability of device registers, internal state logic, and special logic embedded for software/hardware integration.

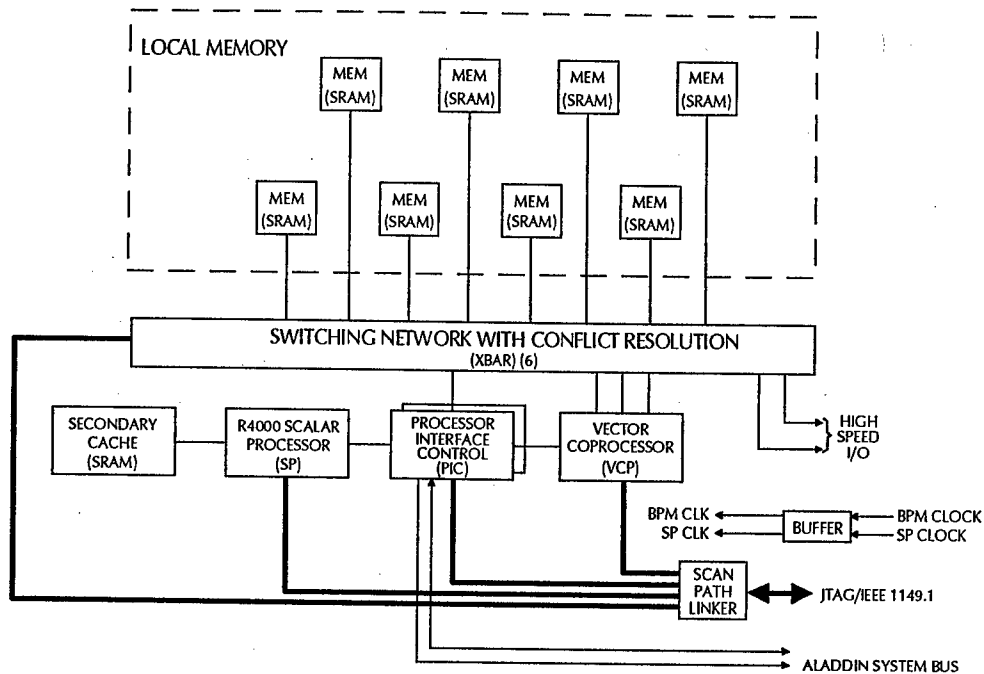


Figure 4.2.4-1 Basic Processing Module Architecture

A significant portion of logic was dedicated specifically for design verification and software test and integration. Table 4.2.4-1 summarizes the test logic used on a BPM. Test logic was controlled via the IEEE 1149.1 test bus and included ASIC internal scan and built-in emulation. Built-in emulation logic in the ASICs provided control of processor execution such as RUN, STOP, SINGLE-STEP, EXAMINE/MODIFY internal registers, and real-time non-intrusive breakpoints and execution trace. A special integration test system was developed for software integration to implement a user interface, debugger, and control the built-in emulation logic via IEEE 1149.1.

Table 4.2.4-1 Aladdin BPM Test Logic Summary

	Device Type				Module
	XBAR(6)	VCP	PIC(2)	R4000	BPM
# of Gates	65,000	165,000	125,000	?	805,000
# of Test Gates	16,500	14,000	16,000	?	145,000
# of 1149.1 Opcodes	60	81	74	2	
# of Scan Paths	26	37	21	3	238
# of Bits in Boundary	271	253	302	319	2,802

Emulation logic was designed into the VCP, PIC, and XBAR ASICs to support coordinated system wide debugging on and between BPMs. Memory breakpoints and execution trace was implemented in the XBARS to monitor and capture memory

transactions. Internal scan was incorporated into the VCP to examine and modify internal software accessible registers. The PIC ASIC implemented the BPM and system level emulation consolidation logic as well as statistics counters for cache misses and various bus transactions. These built-in emulation capabilities, particularly the VCP internal scan, proved invaluable during design verification and troubleshooting.

The Aladdin test bus architecture consists of an IEEE 1149.1 test bus partitioned into sub-rings for each BPM. This test bus architecture was chosen to allow efficient access to a subset of the devices in the cluster and to provide fault tolerance of scan path faults during manufacturing, integration, and test. Each BPM is connected to other BPMs to form one test bus ring. A device called a Scan Path Linker was incorporated on each BPM to partition the R4000, VCP, PICs, and XBARs into four sub-rings. These sub-rings could be collapsed to reduce the overhead of scanning all devices on a BPM and also provided a level of fault tolerance if a sub-ring or a device caused a sub-ring scan path discontinuity. The Aladdin test bus architecture is shown in Figure 4.2.4-2.

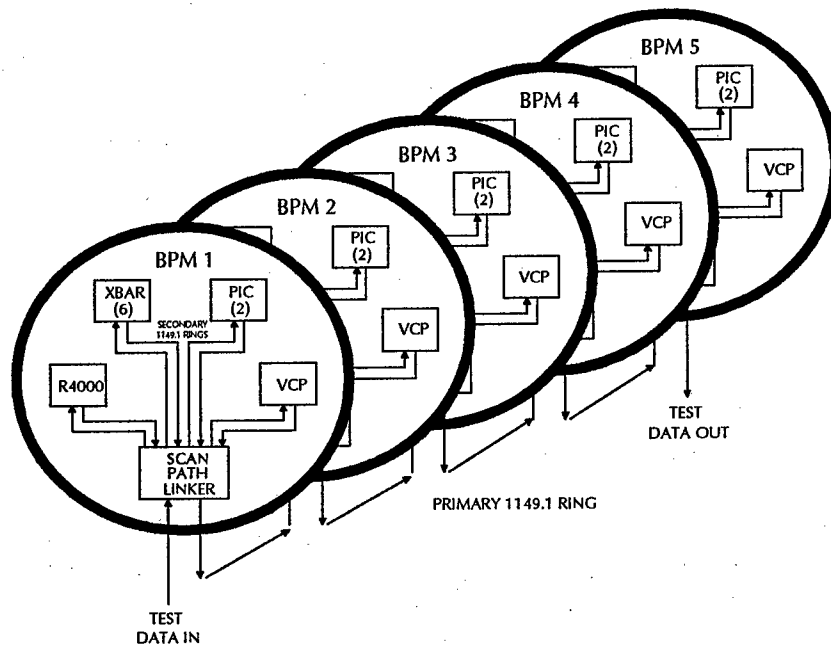


Figure 4.2.4-2. Aladdin Test Bus Architecture.

Other Aladdin physical configurations are being designed to accommodate different weapon systems. One is the SEM-E form factor commonly used in avionics applications. SEM-E modules are approximately 5.9x6.7 inches and less than 0.6 inch thick. Because avionics systems typically have many different module types plugged into a common backplane, they must also share common interfaces. This includes both functional interfaces and test interfaces. To accommodate the test requirements, a SEM-E Aladdin module will have a module level test and maintenance bus (TM-Bus). The TM-Bus provides a backplane-wide test interface between modules to communicate module status

and to provide a gateway to the on-module IEEE 1149.1 test bus. The SEM-E Aladdin test bus architecture is shown in Figure 4.2.4-3.

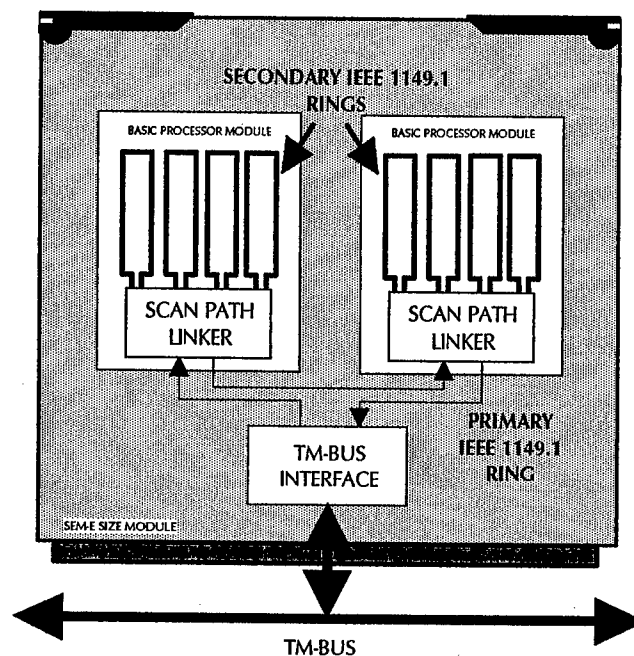


Figure 4.2.4-3. SEM-E Aladdin Test Bus Architecture.

Summary.

The Aladdin program was initially a concept exploration program to demonstrate the feasibility of high performance parallel processing and high density packaging. There were no specific test or test architecture requirements. Therefore all test requirements were derived to meet concept demonstration goals. The result was a hierarchical 1149.1 test bus architecture which was scalable to the number of BPMs in a processing cluster. The easiest way to interface with the unit under test was directly to the 1149.1 test bus, as opposed to a TM-Bus or some other gateway to the 1149.1 test bus. Also, since BPMs were simply plugged together to form a "stack", the 1149.1 test bus always formed a complete ring.

The scan path partitioning using Scan Path Linker devices was useful during initial hardware integration because checkout could continue even if ICs were missing or bad. With the Scan Path Linker devices the 1149.1 sub-rings could be opened or closed in any order to work around open sub-rings caused by bad or missing ICs. This was less of an issue for manufacturing tests, but tests could be executed faster by using the Scan Path Linkers to collapse the 1149.1 test bus so that only the devices under test were being scanned.

4.2.5. Solid State Recorder (SSR).

The Solid State Recorder (SSR) program demonstrates the use of high density, low power solid-state memory for avionics and space data recording applications. The SSR program implemented a solid state memory system comprised of multiple 1.2 Gigabit Memory Units (GMU). GMUs utilize Multi-Chip Module (MCM) technology to provide 1.2 gigabits of memory in 10 cubic inches while using only 1.4 watts of power. To achieve this level of density, 3 dimensional packaging technology was used on the memory to create 3-D memory stacks. Each GMU is comprised of 5 Memory Bank Substrates (MBS) and 2 buffer substrates. The buffer substrates contain I/O buffers for interfacing with other SSR circuitry. The memory bank substrates contain 2 Memory Driver Unit (MDU) ASIC devices and 16 3-D memory stacks. The MDU ASICs provide an interface between the main SSR system bus and the 3-D stack memory. A GMU is shown in Figure 4.2.5-1.

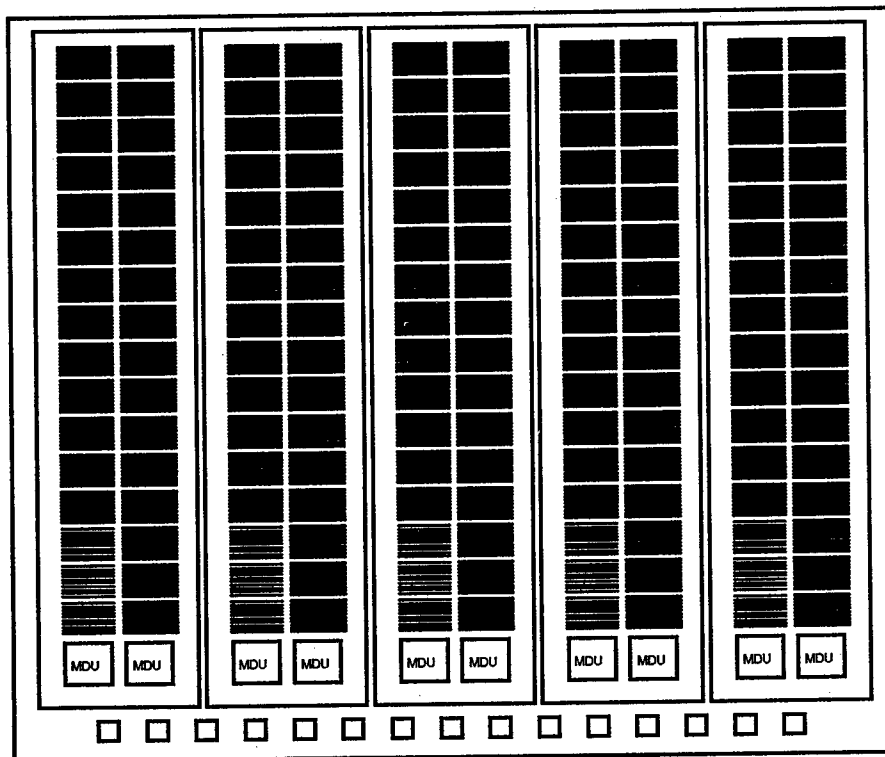


Figure 4.2.5-1. GigaBit Memory Unit

Due to the lack of physical access, IEEE 1149.1 boundary scan was designed into the MDU ASICs to provide pin I/O control to test the memory stacks. The SSR design used boundary scan for design verification, integration and test, manufacturing defects test, and troubleshooting. Scan-based tests consisted of scan path tests, interconnect tests, and functional scan-based tests. The functional scan-based tests were similar to normal functional tests except that they were driven at the I/O pins by boundary scan. This

technique "emulates" a functional operation (driven by functional logic). Traditional Built-In Test software was used for field test which consisted primarily of microprocessor driven RAM tests.

The test bus architecture of each MBS consists of a single IEEE 1149.1 test bus ring. Each memory bank substrate test bus can be individually accessed via a test board. On the test board are five TI 74ACT8999 Scan Path Selectors (SPS) which partition the IEEE 1149.1 test bus into four sub-rings. Each SPS provides the capability to open or close one of the four test bus sub-rings at a time. One SPS provides access to MBS1-MBS4 of each GMU. Four SPSs are used to access GMU1-GMU4. A fifth SPS accesses MBS5 of GMU1-GMU4. The five SPSs are chained together to form the primary test bus ring. This architecture is shown in Figure 4.2.5-2.

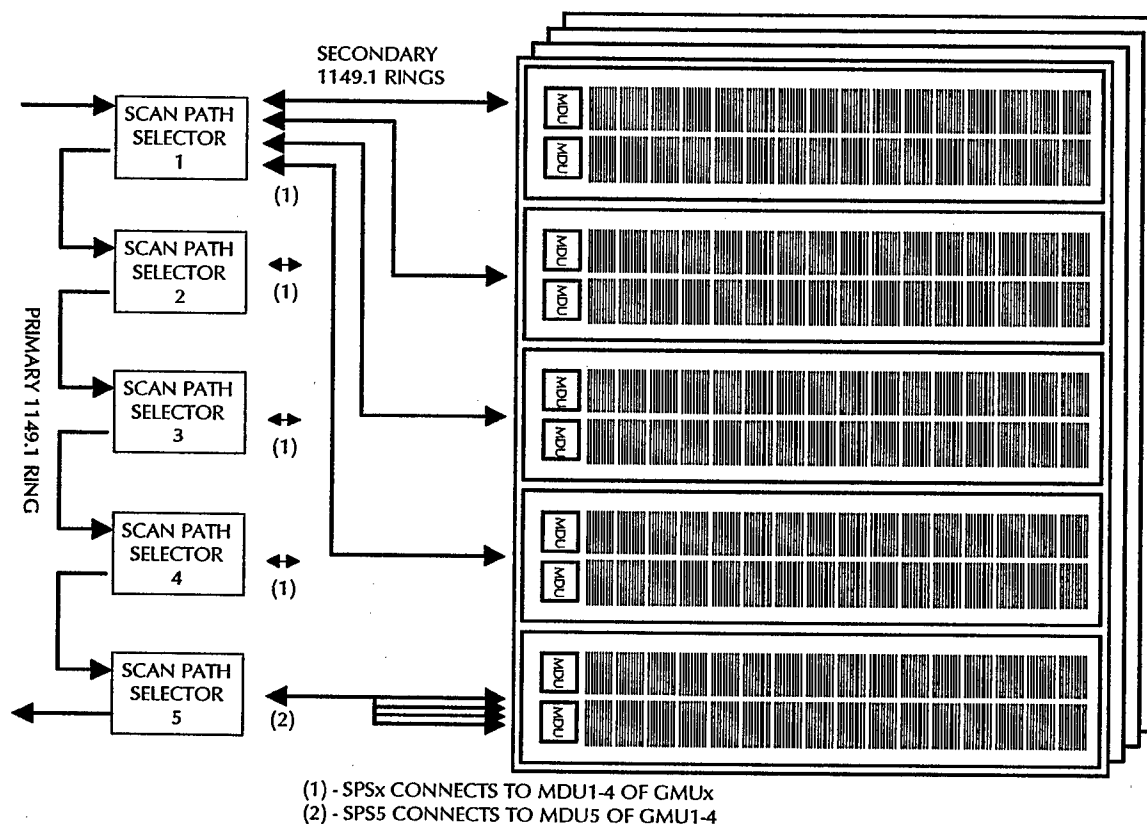


Figure 4.2.5-2. Scan Path Architecture

Summary.

This peculiar scan path architecture was chosen to allow individual MBSs to be accessed individually and independently. The SPSs provide the scan path partitioning so that IEEE 1149.1 based tests can be executed on individual MBSs. This provides a level of fault tolerance during testing and assembly because testing can be performed on individual MBSs while unpopulated or non-functional MBSs can be bypassed. The SSR program was initially a concept exploration program and therefore was limited to lab environments

with access to external equipment. However, as the GMUs are integrated into other systems, the IEEE 1149.1 test bus can be interfaced with an autonomous test controller or a higher level bus such as the TM-Bus.

5. TEST BUS IMPACTS

5.1. IEEE 1149.1 (JTAG).

Because all test buses are implemented in IC silicon, ICs will naturally have mostly negative impacts due to gate count, power, pins, etc. Virtually all negative impacts occur during or resulting from the design phase of ICs, because the design time, gate count, pin count, yield decrease and performance impact are greatest here. However, the negative IC costs are decreasing as tools and technology advance. For boards or modules, the cost of adding 4 or 5 pins is only unfavorable compared to adding no test pins; but no test pins implies no testability. Beyond the design phase, test generation, test application, system debug, hardware/software integration, troubleshooting and repair time show the tremendous benefits of adding the test bus. Simple "piece cost" analysis should be avoided when analyzing test inputs because module and system life cycle benefits are not considered, which is where the greatest advantages are realized.

5.1.1. Design.

IC Pins/Package Size. IEEE 1149.1 absolutely requires the addition of 4 pins, TMS, TCK, TDI and TDO. These 4 extra pins are of no consequence if they do not push a design into a larger package. When the number of functional pins plus 4, or 5 with TRST, is less than the number of pins on the planned package, the device simply will have fewer, if any, pins designated as "no connects." However, if the test bus pushes a design from a 24-pin to a 40-pin package or from a 160-pin to a 208-pin package, the pin counts increase by factors of 67% and 30% respectively. The pinout penalty can be viewed also as the percentage of package pins dedicated to the test bus, shown in Table 5.1.1-1. The values in the table assume only 4 test bus pins and do not include TRST.

Board Test Points/Connector Size. Ad-hoc testability methods typically require bringing each test point directly to a connector. Adding 5 test pins in this way adds only 5 test points; whereas the IEEE 1149.1 bus adds at least as many test points as the number of boundary scan cells on the board, plus access to any internal scan paths. For a memory board test case, an ad-hoc testability approach required over 60 extra pins, compared to the 4 pins for IEEE 1149.1.

IC Gate Count. The number of IC I/O pins drives the number of gates in the boundary scan logic. Because each functional pin must have a boundary cell, the majority of the gates for implementing boundary scan are found in the boundary register. Naturally, more pins means more test logic gates, and fewer pins need fewer test logic gates. Thus, devices with high pin count and little core logic have a higher percentage of gates used for test. Whereas, core-bound ICs (small number of pins relative to size of core) have a smaller percentage of gates for test.

Table 5.1.1-1. 1149.1 IC Package/Pin Ratio

IC Package Size	Percent of 1149.1 IC Pins vs. Total IC Pins
24 pins	16.7 %
40 pins	10.0 %
64 pins	6.3 %
100 pins	4.0%
132 pins	3.0%
160 pins	2.5%
208 pins	1.9%

The number of gates for a near-minimum implementation can be predicted. The survey section listed the required blocks as the TAP controller, the instruction register (IR), the bypass register (BR) and the boundary register. From this information, a formula to estimate number of gates can be derived:

Total IEEE 1149.1 Gate count =

$$(TAP) + [(IR) * (IR \text{ bit width})] + (BR) + [(\# \text{ I/O pins}) * (\# \text{ gates/pin})]$$

Note: Remember to count only the number of functional I/O pins, not power, ground or unused pins.

Given the test cell gate counts for the targeted library and the design variables, the formula yields accurate results. Table 5.1.1-2 shows test cell gate counts for one ASIC vendor's 1.0-micron libraries. Using the table data and the equation, Figure 5.1.1 plots test overhead percentages for standard-cell designs.

Table 5.1.1-2. ASIC Cell Gate Count

IEEE 1149.1 Function	Approximate Number of Gates
Test Access Port (TAP)	~170 gates
Instruction Register	~20 gates/register bit
Bypass Register	~8 gates
Unidirectional Boundary Cell (Standard Cell - Hard Macro)	~15 gates/pin
Bidirectional Boundary Cell (Standard Cell)	~19 gates/pin
Unidirectional Boundary Cell (Gate Array - Soft Macro)	~24 gates/pin

Note: Actual counts will vary between different ASIC libraries.

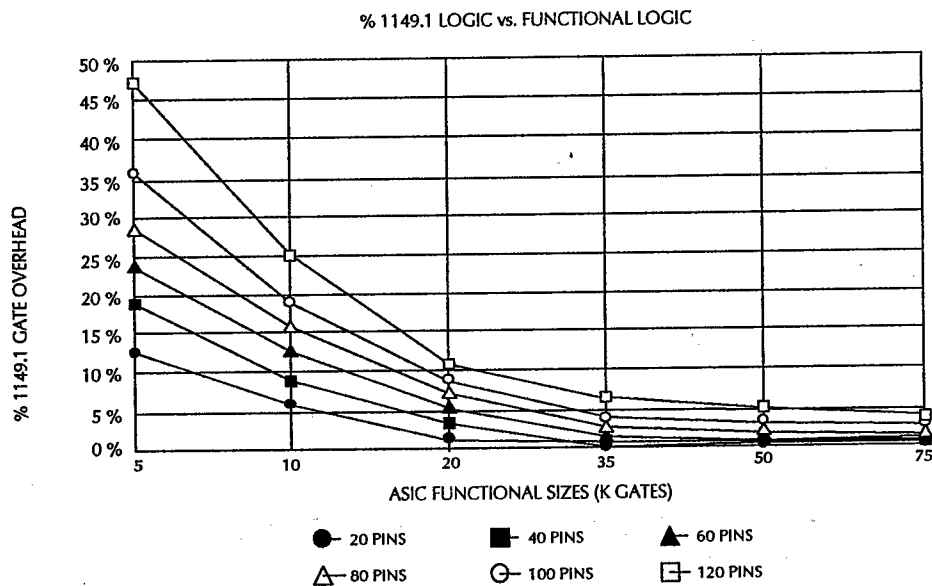


Figure 5.1.1. Standard Cell ASIC 1149.1 Overhead

Table 5.1.1-3 shows gate counts for 3 manufactured ASICs. The first ASIC demonstrates a high test logic percentage expected for an I/O bound device. The first

ASICs library lacks a hard macro, thus requiring the use of a soft macro. (Hard macros are ASIC cells that are "pre-routed" or laid out and optimized for their collective function.) Before declaring 52% too costly, consider two factors. First, this device interfaces a processor to memory. In this location, the boundary scan partitions the processor from memory, enabling separate tests. Second, gate arrays are fixed size, in this case 6K gates, so the IEEE 1149.1 blocks used empty real estate. The second and third examples show progressively more core-bound devices with correspondingly lower test overhead. Their ratio of test logic drops also by use of hard macros available in this standard cell library. Notice that these test overheads are low in spite of adding BIST to allow the devices to self-test internal functions.

Table 5.1.1-3. Overhead Examples of Implementing IEEE 1149.1 and BIST

ASIC Size (gates)	IC Funct	Total I/O Pins / Bidirects	1149.1 prediction ¹	1149.1 actual	BIST	Test Total	Test as % of total
4,753 ²	2,263	70/64	2,470	2,490	0	2,490	52.4
20,000 ^{3,4}	18,350	70/16	1,648	1,650	1,000	2,650	13.3
87,000 ⁴	84,262	70/32	1,712	1,634	1,104	2,738	3.1

Notes:

1. Predicted values are based on the formula, 8 IR bits and data from Table 5.1.1-2. For these devices, the size of the TAP, which has since been optimized, was ~183 gates.
2. Implemented in gate array. Assume ~28 gates/bidirectional pin for prediction.
3. Implemented in standard cell.
4. All gate counts for this ASIC are estimated.

Board Real Estate. The story of board real estate is similar to that for test points and power. While individual IC footprints may grow, the number of devices specifically added for test shrinks. Adding ICs for test, in an ad-hoc fashion, generally does not improve testability as well as adding boundary scan either. Also, routing the bus signals between components can take less space than adding toggle pins or routing many test points from internal nodes directly to edge connectors.

In a memory board test case (mentioned under board test points), ad-hoc test methods replaced 4 components and added 7 more components, while boundary scan test replaced 5 components and added 1 more component. Thus, the ad-hoc approach required 6

more devices to be added for test than did boundary scan. In spite of the higher part count, the ad-hoc board had worse fault isolation. The largest ambiguity group for the ad-hoc board was 4 devices, while for the scannable board it was only 2.

Unlike the example above, some systems require more advanced testability features, such as an embedded test bus controller or a monitor device. Obviously, adding components increases area. However, these additions will tend to be small in part count and will be made only when testability requirements demand them. The test functions they provide would be hard, if not impossible, to duplicate with ad-hoc test methods.

Power. Power dissipation varies according to technology and gate count. CMOS technology draws more power as the amount of active logic or clock frequency increases. In normal operating mode, the only active test logic is the output mux in boundary cells which provide controllability. As a result, the test circuitry is mostly inactive or static and, thus, consumes little power. However, in bipolar or GaA technology the power increase is proportional to the additional gates. Board power is proportional to the sum of all IC power consumption. While, boundary scannable devices can consume more power, fewer devices are needed for testability, possibly lowering the total.

Power data is available for the TI SCOPE octals. To replace non-test octal parts, the test octals must retain high drive without increasing power dissipation. Consequently, TI designed them in BiCMOS technology, combining low-power CMOS in the core logic and high-drive bipolar output buffers. The power consumption of the BCT8244 test octal was compared to that of the BCT244, the FCT244A and the F244, from 0 to 33 MHz. For all frequencies, the BCT8244 used less power than either the FCT244A or the F244. For frequency greater than 4 MHz, the BCT8244 test octal dissipates less power than the BCT244.

Cycle time. Cycle time for the design phase starts at conception and continues until release for manufacturing. The primary tasks are *specification*, *design* and *verification*. Because the IEEE 1149.1 test architecture is so structured and well defined, writing a *specification* for test is much easier than for ad-hoc testability. Board test is largely covered by EXTEST. The number of test pins is known in advance, before manually identifying test points for control and observation. IC test can be accomplished through INTEST, RUNBIST and/or internal scan functions. Manually adding partial internal scan to achieve a high fault grade can be a time-consuming process. Fortunately, identifying nodes for partial internal scan is becoming automated. The same IC test features *specified* for chip test can be reused for "pins-in" test, when the device is mounted on a board or module.

The impact to IC *design* time can be anything from 5 weeks to under an hour. The case which took 5 weeks included time for the designer to learn IEEE 1149.1, specify the implementation, manually *design* the gate level logic and verify the test *design*. In that environment, time to specify, *design* and verify dropped to 3 weeks without the learning curve. When the process is automated, the engineer's time is limited to creating a pin list or writing a BSDL file. Even when a test synthesis program providing automation runs slowly, the labor to execute it is minimal. Depending on its complexity and the amount

of CPU time dedicated to it, the program could run for hours, but it only takes minutes for the engineer to setup and initiate it.

During board level *design*, implementing testability for non-scan components will need the most work, because test logic for boundary-scan is concentrated in the IC *design* effort. When fewer 1149.1 devices are on a board, interconnect testing is harder. However, simply replacing 8-bit buffers, registers or transceivers, common to many boards, with test octal counterparts, helps by partitioning the logic into more testable blocks.

Any special test features or circumstances alter the impacts of the test bus on board design. Some examples are autonomous testing, long scan chains and at-speed testing.

Autonomous Testing. If a board or system is to use scan during autonomous testing, it must control the test bus internally. To do so requires an embedded controller. As indicated in the board real estate section, one device can fill this function. Depending upon whether the system has a microprocessor or not, influences the type of controller. Most off-the-shelf 1149.1 controllers require a host processor for direction. If no host processor is available, the controller must run independently. Because such a controller-processor combination is not a commercial product, it must be *designed*, thus lengthening cycle time. However, supporting autonomous testing without the test bus could be far more difficult.

Long Scan Chains. In very dense boards or systems having exceedingly long scan chains (such as multichip module projects), a scan path linker chip may be used to break the ring into several smaller more manageable rings. Understanding and applying a linker device can take time.

At-Speed Testing. Some systems require on-line, concurrent, non-intrusive, at-speed testing. This can be done using IEEE 1149.1, but, again, impacts the design schedule. See the real-time event qualification test bus extension and notice the monitor chips in the appendix.

Even though digital simulators are getting faster, gate and pattern counts are skyrocketing. One potential benefit of the test bus during IC design *verification* is derived if some internal scan is used. Through internal scan, the designer can set up initial conditions to test a block of logic, and possibly bypass a long test sequence through another block. Besides saving time by shortening the test, the engineer avoids problems caused by the other block being incomplete or not functioning. On the flip side of initialization, expected results on internal nodes can be brought out through TDO, simplifying *verification* of block functionality. Internal scan in effect partitions the chip into smaller blocks which can be tested independently. Additionally, the resulting test could run on a tester without modification.

Today, boards or systems are *verified* often after the first hardware prototype is ready, rather than by simulations. Developers, then, must prove correct manufacturing, similarly to manufacturing test, before functional *verification* or debug can begin. Two steps accomplish the task. First, the scan chain integrity is tested with a flush test. If it fails, test bus signals may be manually probed. Compared to troubleshooting the entire board,

the test bus has limited interconnections, is controlled easily and makes expected value prediction (even of intermediate values) quite straightforward. Once the scan chain is verified, the EXTEST instruction can *verify* much of the board's functional connections and part placements. Now that the board is structurally correct, does it work? By partitioning the functions and adding many virtual test points, both through boundary scan and any IC internal scan, the test bus speeds up functional test generation and debug. As for ICs, functional blocks on a board can be tested independently, when separated from surrounding logic through scan. Divide-and-conquer significantly shortens test creation. Partitioning enables *verifying* remaining areas of the board or system, when a known-bad block exists. For a description of this example, see the Aladdin application section. When functional tests fail, locating the fault is much simpler. Virtual probing can be done without multiple logic analyzers, without manually moving probes from place to place and without saving data from reruns of the same test to get a complete picture. In systems with processors, hardware and software must be integrated. The embedded design debugger extension speeds up integration by enabling all the features associated with a processor emulator. The Aladdin project also successfully used this extension. A section under Test Bus Extensions fully explains the debugger features.

Propagation delay. Propagation delay, a.k.a. path delay, is the time it takes for a change on a logic block input to propagate through a block path to an output. (A block is a black box which may contain one transistor or an entire system.) Delays added to any path in the circuit's functional logic block are of greater concern than the speed of the test logic. In a minimum application of IEEE 1149.1-1990, test affects normal data paths only of signals driving the level or strength of output pins. During EXTEST, the test bus controls these signals through the 2-to-1 output mux of attached boundary cells. However, during normal operation, functional signals also pass through these muxes. Then, the delay through the 2-to-1 mux must be minimized. Propagation delays for cells in current ASIC 1.0 micron CMOS standard cell libraries are ~0.8ns for a hard macro, ~1.8ns for a soft macro and ~200-400ns for a custom cell. Custom cells can be created by building the 2-1 mux into all I/O buffers of the ASIC library. The buffer and mux design prevents bypassing the mux, so that timing is the same with or without test. If the first and only load on an input is a flip-flop data input (often a retiming flip-flop used for synchronization), the input can be controlled without a mux, adding no delay to the functional circuit.

5.1.2. Manufacturing and Test (Fault Detection & Isolation).

IC purchase costs. Boundary-scan ICs, both ASIC and off-the-shelf, cost more to build and purchase than equivalent components without test. Naturally, extra gates and pins are not free. For ASICs the impact will be greater if either the extra pins force the device into a larger package or the extra gates or I/O pads require a larger die. Even when the same package and die sizes are used, more pins must be bonded and more gates must be created. Data for one type of commercially available boundary-scan device, the TI SCOPE octals, puts the cost at roughly 3 times their non-test counterparts. Because of expanded test instructions and the simplistic nature of the normal function, the overhead for test is high and drives up the price.

Board production costs. The cost of building a board prior to testing results from materials and labor, both of which go up with complexity. The cost delta between ad-hoc and boundary-scan test varies from board to board, ranging from cost savings to a higher expense. As shown, ICs without the test bus are cheaper to buy. However, reducing the number of test parts can offset this differential to such a degree, that the price actually goes down. Also, decreasing the number of pins in the test port might mean smaller or fewer connectors, less routing on the board and definitely fewer connector pins to wire. Regardless if the board costs more, the expected savings come during test and maintenance.

Yield. Implementing IEEE 1149.1 affects yield at all levels, IC, board and system. During IC production, yield is measured for each of 3 major steps, sort yield (good die/wafer), assembly yield (good packaging) and final test yield (burn-in testing). Sort yield will drop as gate count increases and assembly yield drops as pin count increases. Burn-in testing is perhaps most affected by how the device timing changes over temperature. Therefore, the slight performance degradation caused by the test control mux may lower final test yield. For board level manufacturing, yield may actually increase as compared to ad-hoc test, if part and pin counts decrease. Thus, the same elements that drive production costs also influence yield. System yield follows board yield proportionally.

Automatic Test Equipment (ATE) issues. For a unit under test (UUT) with IEEE 1149.1 boundary-scan, the test equipment can be as simple as a PC with special software and a hardware interface to the test bus. PC-based testers are inexpensive and portable. While they can be used during design verification and in the factory, the portability and simple connector make them ideal for field maintenance. Not only are they portable by their size, but they are also portable among different types of UUTs, simply by using a different test program. Because these testers only interface with the test bus, no special fixture is needed for the UUT.

PC-based testers are not restricted to standalone operation. They can be combined with traditional in-circuit or functional ATE. When the 2 testers are combined, the application of serial and parallel patterns must be synchronized. This can be done either by handshaking or a higher level controller to start one and stop the other at appropriate times.

If the PC-based tester is not combined with the ATE, the ATE must handle serial pattern application. Such testers typically apply vectors in parallel fashion, as described in the SVF extension section. Parallel access of the serial bus is inefficient, and, so, tester companies, such as Tektronix and Teradyne, are developing special approaches for serial tests.

In systems with wafer scale integration (WSI) or multichip modules (MCMs), in-circuit test (ICT) is uncommon, but circuit densities require testability. Even when circuit nodes are accessible to ICT, eliminating ICT through boundary scan may provide savings. In-circuit testers can range up to \$350,000 with fixtures from \$3,000 to \$12,000 each. Building the fixtures, which often can be done only after a board is debugged, consumes up 3 to 6 weeks of time-to-market.

Test Generation. IEEE 1149.1 shines here. All those virtual test points, which are simple to control and observe, cut test generation time to shreds, while achieving higher fault detection and better isolation than ad-hoc testing. Test engineers can expect detection of all types of faults, opens, shorts to power or ground and adjacent pin shorts, both on output and on input pins. Detecting faults on input pins, even with in-circuit test (ICT), was difficult, because the fault had to be propagated through the device to be observed on an output pin. Boundary-scan puts a virtual test point right at the pin. In the past, detecting a fault was hard enough, but writing a test to locate the source of the fault to a few components was even tougher. Boundary-scan, however, can often identify not only the faulty component, but even its failing pin.

One reason board test generation is so much easier and faster is that the core function of scannable devices can be ignored. In fact, board tests can be created before the ASICs are completed. IEEE 1149.1 can also simplify testing the internal logic of ICs, which are mounted on a board. The test bus enables testing ICs either by reapplying the IC manufacturing tests through INTEST or by executing a RUNBIST instruction. The board test engineer does not need detailed knowledge of the component for either of these test approaches.

Although the boundary scan standard was driven especially by circuits, such as wafer scale integration (WSI), multichip modules (MCM), fine pitch packages, conformally coated boards, surface mount (SMT), etc., which cannot use ICT, boundary scan is also a great boon to ICT. ICT depends on component models to drive output pins and to observe values through input pins. The more complex the device, the longer it takes to write and debug the model. Also, as mentioned, detection of faults on input pins, using these component functional models, is poor. When devices incorporate the 1149.1 standard, writing models and tests for ICT are simple and rarely, if ever, need to be debugged.

Test Application. It would seem that applying tests serially would be slow. However, the shortness of the tests, enabled by ready access to the virtual test points, more than offsets time to shift vectors in and out. Also, time-consuming manual probing to locate faults is bypassed. If the board cannot be probed, the smaller ambiguity groups save time and money, because fewer parts are scrapped and the technician can repair the board faster.

5.1.3. Field Support and Maintenance.

Test Generation and Test Size.

Field tests can be reused from factory tests, and will be especially convenient if the factory tests run on a portable PC-based tester. If built-in test (BIT) and built-in self-test (BIST) are relied on in the field, special considerations should be made. Boards and systems usually have limited memory for storing BIT patterns and limited time to execute BIT. Therefore, direct application of IC manufacturing vectors through INTEST may not be acceptable. An optimum subset of those vectors can be identified to be used for BIT. BIST circuitry or BIT code can be used to generate patterns and compress data. If EXTEST patterns are 'precompiled' rather than interpreted, their execution time can be reduced an order of magnitude.

Reliability.

The IEEE 1149.1 test architecture is required to not affect the normal functionality of the core logic. This requirement reduces the reliability impact of the scan-based logic to only the 2-to-1 multiplexer in the functional data path. This 2-to-1 multiplexer accounts for 2 gates per boundary cell. In a 6000-gate ASIC, with 132 functional pins, the 2-to-1 multiplexer would result in a 4.4 percent gate count increase in the functional data path. This increase, however small, must be weighed against the overall increase in structured testability at the system, board, and device levels. Table 5.1.3-1 compares the features, including reliability, of a standard octal part with a testability octal part. Failure rates listed in the table for the testability parts include all logic, not only the functional gates and the 2-to-1 mux added to the functional path.

Table 5.1.3-1. Feature Comparison for Standard Octal and Testability Octal Parts

Features	Standard Octal Parts	Testability Octal Parts
Pin count	20	24
Gate count	<100	~800
Failure rate: at 0 deg C at 60 deg C	0.0401 0.1500	0.0563 0.2762
Normal functions	Buffer, latch, transceiver, register	Buffer, latch, transceiver, register
Internal test functions for testing of other parts	None	Signature analysis, Pseudo random pattern generation, Boundary scan, Readback and latch (245), I/O toggle mode
External test purposes	Readback latch, Control register	Readback latch, Control register, Pattern generator

While the reliability of an individual IC may decrease, reliability at the board level may improve. As discussed, structured testability, with IEEE 1149.1 boundary-scan, may add fewer parts for test than ad-hoc testability, and thus offset the negative impact on IC reliability. Even if the reliability drops, availability may increase. Consider a system with 200-hour MTBF and a repair time of 10 hours. Its availability is $(1 - 10/200)$ or 95

percent. A more testable system with a 195-hour MTBF and repair time of 5 hours will be available (1 - 5/195) or 97.5 percent of the time.

Table 5.1.3-2 compares reliability among 3 versions of a board, baseline, ad-hoc test and 1149.1 test. Failure rates for tables 5.1.3-1 and 5.1.3-2 were calculated according to MIL-HDBK-217E. Mission failure was determined by recalculating and summing the device failure rates at 60 degrees C for only the mission critical logic. Test logic which does not affect functionality is not mission critical. Mean Time Between Failure (MTBF) is derived from the mission failure rate. This example shows the possibility of part count decreasing from ad-hoc to boundary-scan testability. The higher part count of the ad-hoc board causes increased failures at 0 degree C, where the dominant failure mode is from interconnects. Whereas, the 1149.1 solution edged over the failure rate of the ad-hoc board at 60 degrees C, because at higher temperatures silicon process dominates the failure modes. (Note that the connector was counted as a changed part: Actually, the same connector was selected for all 3 boards, but, over 60 extra pins were used for the ad-hoc board and 4 extra pins were used for the 1149.1 board.)

Table 5.1.3-2. Reliability Comparisons.

Reliability Comparison	Baseline Board	Ad-Hoc Board	IEEE 1149.1 Board
Failure rate at 0.0 deg C at 60.0 deg C	3.4135 36.8410	3.7105 37.8677	3.5670 37.8744
Mission failure (FM) f/mh	20.1273	20.7891	20.7207
MTBF hours	49,700	48,100	48,300
Part count (changed / added)	18	25 (4/7)	19 (7/1)

Maintainability.

The overall impact of IEEE 1149.1 in the maintenance environment is as follows:

- enables more compact/portable flight line test equipment
- provides better isolation of failures to boards and individual devices
- provides for continuity of test vectors/test software between factory, field, and depot

- shortens troubleshooting and repair times as a result of improved isolation with its reduced ambiguity and "hands-off" virtual probing
- increases availability as maintenance time decreases

All of these features reduce maintainability expenses over the life of a design.

5.1.4. Cost Summary.

Costs can be compared in three ways, no testability, ad hoc test and use of IEEE 1149.1-1990. Ad-hoc costs are not easily quantified, as they can vary widely, but their general drawbacks and benefits can be contrasted with the test bus, as in Table 5.1.4.

Table 5.1.4. Differences between Traditional (Ad-hoc) and Boundary-Scan Methods

Tradeoffs	Traditional	Boundary-Scan
Approaches	Usually not structured Usually starts after design has gone to manufacturing	Structured approach Starts during design
Test Equipment	Complexity/density of designs require more test equipment to test and debug systems All "Hands On" Hot mockup, logic analyzer, oscilloscope, ...	Limited equipment needed to test and debug systems. No need for expensive special test equipment during production test, depot test, etc... "Hands On" limited PC-AT software, ohmmeter, test octals
Software	Tests generally not reusable	Reusable test software goes from debug--board--production ...
Internal Visibility	Fault isolation depends on ad-hoc testability Manual probing required Required external hardware to look at internal nodes	Fault isolation increased No manual probing required Have internal node visibility
Real Estate	Additional real estate required to support ad-hoc testability and space to allow for probing	Additional real estate, four I/O pins, and a larger footprint IC

5.2. IEEE P1149.5/TM-Bus.

5.2.1. Design Considerations.

In the design of a module compliant to IEEE P1149.5, there are several considerations that must be addressed. As with any system design, the system-level considerations are addressed first as a part of the system requirements definition process. Some system-unique requirements will impact the IEEE P1149.5 requirements for a module fitting in the system. The goal is for system-specific profiles to be available to both commercial off-the-shelf equipment and military equipment vendors. The IEEE P1149.5 system's engineering considerations include:

1) MTM-Bus Physical Layer

- Electrical characteristics
- Timing parameters

Factors involved include the number of modules in the system, the system's backplane characteristics (e.g., length, impedance, capacitance, termination), the MCLK source, hot insertion/removal, etc.

2) Fault Tolerance

- Redundant/Backup Modules (Bus mastership arbitration/transfer)
- Multiple-Bus Architectures (Redundant buses, alternate buses, bus switching, etc.)

There are also several module considerations in the development process. Many of these considerations are influenced by the anticipated target system:

- 1) Interface type (master-only, slave-only, master/slave).
- 2) Slave module addressing method.
- 3) Support/need for Module Pause Request (MPR) signal.
- 4) Fault tolerance (packet count, MPR timer, application specific bus errors, etc.).
- 5) Error handling (for incorrect packet count, port transfer error, parity error).
- 6) Accessibility to module resources (i.e., data transfer ports).
- 7) Register(s) definitions (Module status register & other status registers).
- 8) Module unique functions (user-defined commands and message formats).
- 9) Level of support for:
 - Module Interconnect Control and Test commands,
 - Module Initialization and Self-Test commands,
 - Data Transfer commands.

5.2.2. Fault Detection/Isolation.

The IEEE P1149.5 MTM-Bus provides no real fault detection or isolation for a module. The bus provides a standard interface to module test resources for control and observing status. It supports fault reporting through its standard status registers and the MTM-Bus interrupt capability.

The MTM-Bus does provide fault detection/isolation support for module-to-module interconnects, much like IEEE 1149.1 does for IC-to-IC interconnects, through the recommended Module Interconnect Control and Test commands. These commands ensure that each module input and output can be controlled and observed via the MTM-Bus such that a static module-to-module interconnect test can be performed.

The MTM-Bus does provide significant fault detection for the bus physical, link and message layers. The detectable faults include:

- 1) Single-bit packet errors (detectable via parity).
- 2) Bus signal stuck-at faults. This is provided for MMD, MCTL, and MSD signals through collision detection, packet parity, detection of illegal commands, and state sequence errors.
 - MMD or MSD signal Stuck-At 1 => Illegal command, Bus collision
 - MMD or MSD signal Stuck-At 0 => Parity Error, Bus collision
 - MCTL signal Stuck-At 0 => State Sequence Error, Bus collision
 - MCTL signal Stuck-At 1 => State Sequence Error, Bus collision
- 3) Message length errors (supported via packet count).
- 4) Bus protocol errors (detectable via state sequence error for MCTL and MMD; detectable via data overrun error for MPR).
- 5) Command sequence errors for critical module control commands.
- 6) Message content integrity can be verified through:
 - Detection of Illegal Port Selected (e.g., attempted to access a port that did not exist or there was a potential module addressing problem)
 - Detection of Illegal commands (e.g., attempted execution of reserved opcodes).

MTM-Bus testing and (maximum module fault isolation) is supported through the Data Echo command. The Data Echo command provides a packet-level wraparound feature at the slave module's interface. This capability allows the bus to be fully exercised while relying on a minimum amount of the module's interface logic.

5.2.3. Cost Summary.

There are both hardware and software costs associated with an IEEE P1149.5 module interface.

The hardware cost, of course, depends upon the requirements/options selected through the design considerations mentioned earlier, and the implementation of those requirements/options. The cost of a P1149.5 slave interface is typically more than that for a master due to the design complexity and the fact that more hardware/software trade-offs can be performed for the master design. Assuming an Application Specific Integrated Circuit (ASIC) implementation for a majority of the module's P1149.5 slave interface (Physical layer, Link layer, and a large portion of the Message layer), the cost in terms of 2-input NAND gate equivalents may range from:

3000 gates for a slave meeting the minimum P1149.5 requirements, up to

18000 gates for a complex slave with additional fault tolerance (dual bus architecture, mastership arbitration, module direct memory access, etc.).

There are significantly more tradeoffs that can be made for a master implementation due to the minimal master requirements imposed by P1149.5. Assuming an ASIC implementation of only the Physical layer and Link layer, the cost in terms of 2-input NAND gate equivalents may range from:

2000 gates for a master with a simple CPU interface, up to

15,000 gates for a complex master with additional fault tolerance (dual bus architecture, mastership arbitration, MPR timer, module direct memory access, etc.).

The software cost for a P1149.5 MTM-Bus implementation is primarily related to the Message Layer. This software cost may vary greatly depending on the hardware/software trade-offs selected during implementation. Again, the software impacts are greater for a slave module than a master module, assuming that only the Message Layer is implemented in software. For a master module, the Message Layer is primarily message/packet construction, memory block moves to transfer/receive packets from the Link Layer, and message error handling. The Message Layer for a slave module depends heavily on the functions implemented for that module (e.g., accessible resources, user-defined functions/commands, etc.). A slave's Message Layer might consist of command recognition, message/packet construction in response to an MTM-Bus command, interface to the on-module application(s)/resources, etc.

5.3. System Buses.

System buses evaluated during this study were found to impose minimum impact on system design architectures except for the considerations generally performed during system engineering. These considerations include:

- System requirements associated with testing strategy; design verification, factory, field, depot, and maintenance
- Fault detection, fault isolation, and throughput requirements; system, subsystem, module
- Diagnostic data requirements; type and volume of data, and requester
- Hierarchical test initiation and status retrieval by higher test buses

- System partitioning requirements; analog/digital, subsystem/module/integrated circuit
- System penalties associated with interfacing to system buses; added logic, weight, topology, interface modules, etc.

From these system considerations, the value of a dedicated system bus for testing can be determined.

6. TEST BUS EVALUATION RECOMMENDATIONS

The following recommendations are being made to the DOD as a result of the studies performed during the Test Bus Evaluation. These recommendations reflect key areas of concern for the successful implementation of test buses in DoD designs. They are described in two forms, first, as test bus design criteria, and second, as management issues. The design criteria recommendations will address those items critical to a successful test bus system requirements allocation and design implementation. The management section will address those items that are dependent on policies and directives, or impacted by further R&D funding.

6.1. Test Bus Design Criteria Recommendations.

1. Mil-Std 1814 (Integrated Diagnostics Roadmap and Requirements Document) should be updated to require a test bus architecture trade-off analysis, at the system through module level, as part of the Integrated Diagnostics requirements allocation and derivation, and a vertical testability traceability analysis, during which, the use of standard test buses will be evaluated for use in multiple testing environments (e.g. BIT, design verification, factory test, depot test, etc.).

Establishing the requirements for a test bus architecture is the first step in assuring standard test buses are designed into weapon systems. This starts at the Concept/Exploration contract phase and carries through the Engineering and Manufacturing Development. Mil-Std 1814 and the Air Force Guide Specification 87256 go into great detail on defining the total spectrum of the diagnostic process and requirements. These documents need to be updated to include the following changes;

- MS 1814, requirement for a Test Bus Architecture Trade-off Analysis
- MS 1814, requirement for a Vertical Testability Traceability Program
- MS 1814 Appendix A, test bus detailed requirements and verification
- MS 1814 Appendix G, impact of vertical test compatibility from test buses
- AFGS 87256, update 3.x.1.3-3.x.1.x of Segment, Element, Subsystem and Assembly Levels.

2. Mil-Std 2165 (Testability Program for Systems and Equipments) should be updated to include test bus architecture trade-off analyses and detailed design criteria in Appendix B, Inherent Testability Checklist.

Mil-Std 2165 defines the requirements for performing a testability design and assessment. The requirements defined in Tasks 101, 102, 201, 202 and 203 should be updated to reflect the emerging standard test buses and boundary scan techniques. The criteria defined in the next section should be added to the Checklist, as questions, to assure that test bus trade-offs are considered throughout the system design cycle.

3. Further design guidance for both Mil-Stds (1814 and 2165) should include the following criteria as defined for systems, modules and devices. This criteria has been taken from the studies performed during the test bus evaluation and is applicable to any general test bus development effort.

- a) The system test bus architecture should take into account the following criteria;
- Determine the system requirements for the overall weapon system test strategy considering life cycle test requirements such as design verification, factory test, safety, maintenance, depot repair, etc.,
 - Determine what diagnostic information is required (e.g. go/nogo, detailed results, fault log, fault filters, sensor data, etc.) and what are the external interfaces; such as the pilot, the maintainer, another subsystem, down the backplane, etc.,
 - Determine how the system test information will be communicated and what system bus is required based on data, throughput, expansion, etc.,
 - Ensure that the lowest level of test information can be interrogated from the highest level via a hierarchy of test buses (e.g. 1553-to-1149.5-to-1149.1),
 - Determine which module test bus is required based on data, throughput, expansion, size, topology, etc.,
 - IEEE 1149.1, if a small number of modules (<~6), and a fixed system configuration, and/or using "dumb modules"
 - IEEE 1149.5, if a significant number of modules (>~4), with a non-fixed configuration, and/or "smart modules"
- b) The module test bus architecture should further consider the following criteria;
- Consider the module test strategy and the integration of its test features with the test bus capabilities (e.g. BIT software, LRM BIST, fault log, etc.),
 - Determine if an IC test bus is needed based on test requirements, FD/FI, data, throughput, expansion, emulation, etc.,
 - If the designs use ASICs from different vendors or ASICs intended for multiple applications, specify 1149.1 so that test information may be accessible via a common method,
 - If the design is all or partial analog (mixed signal), consider using 1149.1 to partition the analog circuitry from the digital, and to communicate analog diagnostic results to the module diagnostic control function,
 - If the system/module/or IC test bus is not justifiable on its own, consider "added value" features which are useful for related system requirements (e.g. functional messages, hardware or software debug & integration, etc.),
 - There is no need to completely dedicate a system level test bus unless fault tolerance (e.g. for safety requirements) or throughput requirements are critical.
- c) The device test bus architecture should further consider the following criteria;
- When physical access is lost (can't be probed) at a node on a module (i.e. in multi-chip module applications, etc.) boundary scan should be implemented. As a general rule, node depths of greater than five clock cycles should require a boundary scan cell.

- Consider interfacing all embedded device testability features (e.g. internal scan, BIST, Cross-check, etc.) to the IEEE 1149.1 test bus for reuse at the module and system test levels.
- All custom or VLSI device designs (ASICs, Linear, FPGA, Processors, etc.) should be required to perform a trade-off analysis on the use of and interface to the IEEE 1149.1.
 - If there is a desire to use standard test methods (boundary scan), specify 1149.1,
 - Consider off-the-shelf devices with IEEE 1149.1 over equivalent devices without 1149.1,
 - Consider impacts of fault detection and isolation requirements.

6.2. Management Recommendations.

6.2.1 DoD Policies and Directives.

1. A standard for data and information protocols needs to be established for diagnostic data that is passed across system buses to both the system fault management function and directly to ground maintenance.

A critical issue in systems today is the inability to communicate common diagnostic information between electronic subsystems. In order to assure fleet level trending and logistics analysis for diagnostic maturation, a common set of metrics needs to be specified that support diagnostic analysis. These metrics should take into account fault log data recorded on the module for use in the Depot. At the aircraft system level the standardization of this data is critical for using standard test equipment at the flight line (e.g. IMIS, IFTE CTSIII, portable testers, etc.) to augment and support the two level maintenance concepts.

2. The DoD should promote and endorse the use of selective commercial standard test buses in weapon system developments. These should include;

- IEEE 1149.1 as the device-to-device level test bus,
- IEEE P1149.5 as the core module backplane bus (when it is passed),
- Other IEEE 1149.x standards as they emerge.

The DoD should provide a focal point for defense industry test bus standards for definition, feedback, balloting, etc. The DoD should provide seed funding for studies and developments of new standards within the commercial industry. Unfortunately, many standards lack the financial backing to ever get off the ground. DoD could provide the forum to educate industry, raise confidence that standards would be supported in the near term, and spur other commercial industry to support the standards. This could include silicon suppliers, CAE vendors, test equipment manufacturers, and others. The test community also needs standard metrics for the communication of test bus implementation benefits (i.e. quality, cost, productivity, cycle time, etc.).

3. Each contractor should be required to use the defacto commercial standard test buses (e.g. IEEE 1149.1, P1149.5, etc.) as established within the weapon system test bus architecture or else provide justification why not.

With today's highly integrated technologies DoD has already come to the realization that standards best work if supported across the entire electronics industry. By joining forces with the commercial sector a greater amount of funding can be brought to bear on the institutionalization of the standard and its supporting products. As systems become more and more integrated and are designed with large portions of multi-vendor components, the need for standard test buses increases greatly. Military systems have advanced to the point where most systems require test buses and, therefore, a mandate to use standard test buses will ensure implementation unless valid justification exists to warrant an exception.

4. DoD standard ATE programs (e.g. MATE, F-22 CATS, CASS, IFTE, etc.) need to support common access to standard embedded test buses and the diagnostic results contained on the module and subsystem.

The conventional wisdom behind each standard Automatic Test Equipment (ATE) built in DoD has been to ignore the embedded diagnostic capability of the unit under test. It is essential to the success of 2-level maintenance that the Depot ATE use the standard test buses to interface to module BIT logs and for exercising additional test capability, such as ASIC BIST. It is also imperative that the production test in the factory utilize these capabilities to increase yields, cycle time and decrease ATE costs. The F-22 Common Automatic Test System (CATS) is providing an interface to both IEEE 1149.1 and the IEEE P1149.5 which will provide a critical link from the Field to the Depot. Whereas the Army's IFTE and the Navy CASS ATE do not yet provide such interfaces.

5. Current AF thrusts in Test Program Set generation tools do not leverage embedded module techniques (such as IEEE 1149.1 and boundary scan). These programs (Virtual Test, PAP-E, ...) need to be briefed on the advantages of Hierarchical Testability.

As in the case above with ATE, Test Program Set (TPS) generation tools have not linked up with the onslaught of standard test bus (and boundary scan) automation tools hitting the market. These tools provide the control to the test bus and the generation and verification of test vectors. New technology starts in TPS tools (such as Virtual Test) need to understand how the requirements based testing (typically analog in nature) interfaces with vector based testing (typically used for digital) and what role the standard test bus architecture plays in these tools.

6. DoD needs to decide on which standard data structures (e.g. EDIF, WAVES, etc.) and control languages (e.g. BSDL, HSDL, SVF, etc.) they will use so that CAE vendors and users can better support these structures.

The standards efforts in test vector data and information structures are essential tools to the goal of tester independence for TPSs. The current standards have overlapping capabilities. Likewise test bus standard control languages have both complimenting and overlapping capabilities. The DoD needs to drive the use of these standards and give guidance to the Defense Industry community on their applicability. Demonstration

programs, such as the AF PAP-E program, should be conducted to further clarify their advantage to weapon system supportability.

6.2.2 DoD R&D Funding Recommendations.

1. DoD should establish a capability for verification of test bus standards compliance and for interoperability of test buses within weapon system developments.

DoD needs to create an ability to assure that test bus implementations meet the specifications. Commercial tools, such as the AT&T TapDance tool for IEEE 1149.1, should be developed to provide weapon system integrators this capability. A procedure to assure the interoperability of test buses should be provided for cases where common modules are to be used across subsystems and where different test bus implementations interface to the next level test bus. For example, it is essential that interoperability is verified where multiple module vendors have used different chip set implementations of the IEEE P1149.5 (and its avionic extensions) that reside on a common subsystem backplane.

2. IEEE P1149.5 avionic (or vetronic) extensions should be defined for each unique application (e.g. master only, slave only, maintenance controller, etc.) so that interoperability between multi-vendor modules is assured.

This is especially critical for the major aircraft developments (e.g. F-22, F-16 upgrades, etc.). The Air Force needs to take a proactive role in establishing the definition of these extensions. This effort is being pursued by the Society of Automotive Engineers (SAE) with a watchful eye from the IEEE 1149.5 committee. However, additional module bus extensions (e.g. TSMD, scannable connectors, etc.) are needed to improve the fault tolerance of critical safety driven systems (such as aircraft guidance systems).

3. An approach, similar to IEEE 1149.1, to the interface and testing of analog circuits, should be championed by the DoD due to the impact on mission reliability from aircraft sensor system failures.

The work being done by the IEEE P1149.4 committee should hold much interest for DoD weapon system developers. High frequency aircraft sensors (e.g. Radars, GPS, IFF, etc.) have become a critical impact to the fault tolerance of avionic systems. Few efforts have been undertaken by DoD to solve the lack of embedded diagnostic capability in these subsystems. One potential solution may be to interface to an A/D with IEEE 1149.1 for the test results. But much more is needed. Just as boundary scan has become the key test approach for high density digital circuitry, so is a structured technique needed for testing RF/IF, electro optics, lasers, etc.

4. A standard test bus simulation model library should be developed based on the implementations and extensions described in this document. This library should include VHDL models on the IEEE 1149.1 TAP, IEEE 1149.5 master, slave, etc.

One of the major cost benefits in using a standard test bus architecture is in the ability to reuse (both vertically and horizontally) hardware and software test algorithms and control structures. The test bus evaluation study has pointed out the advantage in using common chip sets, control software, and test bus languages across multiple designs.

Another area for reuse is to create a common VHDL model library of the core test bus architectures. This library could be provided to designers across the Defense industry assuring a better success of bus interoperability.

5. DoD contractors who use commercial off-the-shelf parts (e.g. devices, modules, subsystems, etc.) that have a standard test bus (e.g. IEEE 1149.1, 1149.5, etc.) within them, should be given incentives by the DoD.

The larger electronics companies are beginning to see the advantage to using standard test buses in their manufacturing areas. However, the perception that added test circuitry is a cost burden still persists within the design community. This reflects the old "piece cost" mentality and does not consider the module and system benefits. IC/ASIC vendors will not add test unless it is required by the customer, yet the customer (the weapon system integrator) is not given incentives by the DoD. Requirements alone may not be sufficient motivation to assure the best test approach. Financial incentives could be a stronger motivator to accelerating the implementation of standard test bus architectures in today's designs.

6. The DoD needs to establish a training and education program focused on the benefits and design of standard test buses.

The commercial community (e.g. Self-Test Services, TI, etc.) has already developed some excellent courses in the application of IEEE 1149.1 and associated products. The DoD needs to look at education of system and module test buses as applied in the defense industry environment since the above courses are targeted for the commercial IC developer. The ownership of the training and education process is likely the fastest way the Services could institutionalize the use of standard commercial test buses in their products.

7. The AF should take the lead to develop a lessons learned data base on all standard test bus applications and boundary scan implementations so that the use of commercial test bus standards are accelerated.

The Test Bus Evaluation is a good start to such a data base. As indicated by the TBE study most of the current lessons learned will come from the IEEE 1149.1, since it has been in existence for some time. However, there have been many implementations of the module level TM-Bus that could be recorded. This lessons learned data base could be a valuable asset to a Defense sponsored training course on the use of standard test buses.

8. Further DoD investments for studies in boundary scan techniques and developments of automation tools are needed in the following areas;

- Diagnostic algorithm development,
- Test vector generation (e.g. ATPG, etc.),
- Data compaction and compression techniques,
- Test program generation,
- Analog/RF sensor test bus interfaces.

SUMMARY

It should be noted that significant issues exist in test and diagnostic verification. Most DoD programs require quantified FD/FI percentages (e.g. 98% detection), but the implementor is left to his/her experience to design to that requirement. Additionally, FD/FI verification is performed manually, which is subjective. However, if specific test requirements are specified, such as the test buses and the DFT techniques required, then all implementations will share a common set of test capabilities, regardless of the engineering design experience applied. The above recommendations are just a start in the pursuit to achieve this goal.

APPENDIX A.

COMMERCIALLY AVAILABLE TEST BUS PRODUCTS

The following tables list products supporting test bus standards. Note that the majority of support is for the IEEE 1149.1 test bus. To qualify for publication in this appendix, products must have been available by the end of 1992. Certainly, there will always be new products flowing from silicon vendors; better instrumentation to manufacture, test, and debug these designs; and new tools to help glue the design and manufacturing flows together. The lists below may not contain all vendors and tools due to the dynamic growth of the industry.

Silicon.

A list of silicon vendors has been provided. These tables are divided into ASIC vendors and catalog product vendors. The catalog products shown are those gleaned from many sources, and many vendors.

Instrumentation and ATE.

A list of Automatic Test Equipment and Instrumentation which has support for the test bus standards is provided. Some of these products will require a hardware upgrade to allow high speed access to the test bus architecture on your boards, while others are driven by software, only.

TOOLS.

A list of software and hardware tools is provided which either stand alone or provide support services for another piece of test equipment. Keep in mind that the tools may be what bind your silicon to your test resources. It is very important that these components "play" together. Database translation software development tasks should not have to be a part of system design, development, and test.

SUMMARY.

In summary, there are, in fact, many vendors supporting the IEEE Std 1149.1-1990 architecture and few supporting the lesser known standards (P1149.5, P1149.2, etc.). There are also many uses for the architecture, once the commitment has been made to incorporate it in your system. However, it is very important that each vendor and tool be investigated so that your overall test bus methodology can be a successful one.

ASIC Foundries with 1149.1 Support.

- Analog Devices
- AT&T Microelectronics
- Atmel
- European Silicon Structures

- Fujitsu
- GEC-Plessey Semiconductors
- Gould-AMI Semiconductors
- Harris
- Honeywell
- Lasarray
- LSI Logic
- Matra
- Mitsubishi
- Motorola
- National Semiconductor
- NCR
- NEC
- OKI
- Philips
- Raytheon
- SGS-Thomson
- Siemens
- Texas Instruments
- Thomson Composants Militaires et Spatiaux
- Toshiba
- United Technologies Microelectronics Center
- Vertex
- VLSI Technology

FPGA Vendors with 1149.1 Support.

- Crosspoint Solutions
- Texas Instruments
- Xilinx

Catalog Parts with 1149.1 Support

VENDOR	PART NUMBER	DESCRIPTION
Advanced RISC Machines	ARM60	RISC processor
Advanced RISC Machines	ARM60	RISC processor
Advanced RISC Machines	MEMC20	Memory Controller
AMD	Am29030	Common Imaging Engine
AMD	AM29035	Common Imaging Engine
AMD	AM29200	RISC microcontroller
Analog Devices	ADSP21020	Digital Signal Processor
AT&T	WEDSP16C	Digital Signal Processor
AT&T	WEDS16J	Digital Signal Processor
AT&T	WEDS1610	Digital Signal Processor
AT&T	WEDS1616	Digital Signal Processor
AT&T	497AA	Boundary Scan Master
Brooktree	Bt463	RAMDAC
Fujitsu	MB86930	SPARC™-based controller
Honeywell	HTIU2000	Interface between VHSIC TM-bus and 1149.1
I-Cube Design Systems	IQ160	Field Programmable Interconnect Device
Integrated Device Technology	IDT79 R4000 PC	MIPS R4000 compatible processor
Intel	i80486DX-50	Microprocessor
Intel	i82490DX	Second level 80486DX cache SRAM
Intel	i82495DX	Second level 80860DX cache controller
Intel	i80860XP	RISC processor
Intel	i82490XP	Second level 80860XP cache SRAM
Intel	i82495XP	Second level 80860XP cache controller
MIPS	R4000	RISC Processor
Motorola	MC68040	Microprocessor
Motorola	MC68330	Microcontroller
Motorola	MC68340	Microcontroller
Motorola	MC56002	Digital Signal Processor
Motorola	MC56156	Digital Signal Processor
Motorola	MC86001	Digital Signal Processor

Catalog Parts with 1149.1 Support - Continued

Motorola	MC88110	RISC processor
National Semiconductor	SCANPSC100FSC	Boundary-scan parallel/serial converter
National Semiconductor	SCAN18245TSSC	18 bit transceiver
National Semiconductor	SCAN18373TSSC	18 bit D-type transparent latch
National Semiconductor	SCAN18374TSSC	18 bit D-type edge triggered flip-flop
National Semiconductor	SCAN18540TSSC	18 bit inverting line driver
National Semiconductor	SCAN18541TSSC	18 bit line driver
Siemens	SAB-R4000 PC	MIPS R4000 compatible processor
Texas Instruments	SN74BCT8240	Octal inverting bus buffer
Texas Instruments	SN74BCT8244	Octal bus buffer
Texas Instruments	SN74BCT8245	Octal bus transceiver
Texas Instruments	SN74BCT8373	Octal D-type transparent latch
Texas Instruments	SN74BCT8374	Octal D-type edge triggered flip-flop
Texas Instruments	SN74ABT8240	Octal bus inverting buffer
Texas Instruments	SN74ABT8244	Octal bus buffer
Texas Instruments	SN74ABT8245	Octal bus transceiver
Texas Instruments	SN74ABT8373	Octal D-type transparent latch
Texas Instruments	SN74ABT8374	Octal D-type edge triggered flip-flop
Texas Instruments	SN74ABT8543	Octal registered bus transceiver
Texas Instruments	SN74ABT8646	Octal registered bus transceiver
Texas Instruments	SN74ABT8652	Octal registered bus transceiver
Texas Instruments	SN74ABT8952	Octal registered bus transceiver
Texas Instruments	SN74ABT18240	18 bit inverting bus buffer
Texas Instruments	SN74ABT18244	18 bit bus buffer
Texas Instruments	SN74ABT18245	18 bit bus transceiver
Texas Instruments	SN74ABT18373	18 bit D-type transparent latch
Texas Instruments	SN74ABT18374	18 bit D-type edge triggered flip-flop
Texas Instruments	SN74ABT18502	18 bit registered bus transceiver
Texas Instruments	SN74ABT18504	20 bit registered bus transceiver
Texas Instruments	SN74ABT18646	18 bit registered bus transceiver
Texas Instruments	SN74ABT18652	18 bit registered bus transceiver
Texas Instruments	TFB2001	Futurebus+ parallel cache controller

Catalog Parts with 1149.1 Support - Continued

Texas Instruments	TFB2021	Futurebus+ datapath unit
Texas Instruments	TFB2050	Futurebus+ parallel protocol cache controller
Texas Instruments	TFB2010	Futurebus+ arbitrated bus controller
Texas Instruments	TFB2011	Futurebus+ programmable central bus arbiter
Texas Instruments	TMS320C40	Floating point parallel processing unit
Texas Instruments	TMS320C50	Digital Signal Processor
Texas Instruments	TMS320C51	Digital Signal Processor
Texas Instruments	TMS320C53	Digital Signal Processor
Texas Instruments	TMS34082	Floating point graphics co-processor
Texas Instruments	SuperSPARC	SuperSPARC processor
Texas Instruments	SuperSPARC	SuperSPARC cache controller
Texas Instruments	TMS29F816	Diary EEPROM memory
Texas Instruments	SN74ACT8994	Digital bus monitor
Texas Instruments	SN74ACT8990	Test Bus Controller
Texas Instruments	SN74ACT8997	Scan Path Linker
Texas Instruments	SN74ACT8999	Scan Path Selector
Toshiba	MB86900	MIPS R4000 compatible processor
TRW	TMC22190	Digital video encoder
UVC	UVC7710	Multimedia Microprocessor
VLSI Technology	VY86C060	RISC processor
VLSI Technology	VY86C061	RISC processor
VLSI Technology	VY86C600	RISC processor

Instrumentation and ATE Suppliers with 1149.1 Support

	PRODUCT	DESCRIPTION	COMPANY
1	BST800	MDA with 1149.1 subsystem for production PCB testing	Brothers Electronics
2	S2000	Functional test system	Computer Automation
3	BA-1149.1	Boundary-Scan bus analyzer for use with HP 1650/10/40 logic analyzers	Corelis
4	CVXI-1149.1	VXIbus boundary-scan controller module	Corelis
5	PC-1149.1	PC/AT bus boundary-scan controller card	Corelis
6	PM 3580	Logic analyzer family	Fluke / Philips
7	9400	MDA with boundary-scan test	Fluke / Philips
8	GR-227x™	In-circuit tester family	GenRad
9	GR-228x™	In-circuit tester family	GenRad
10	HP 3070™ series	In-circuit tester family	Hewlett Packard
11	HP 3065AT™	Board test system	Hewlett Packard
12	XLScan	Prototype test system	IMS
13	ScanView	Software package which allows ASSET™ to communicate with conventional ATE via an RS-232 interface, thus providing simultaneous in-circuit and boundary-scan test capabilities	Intellitech
14	MFC1149.1	Plug-in board for PC	Jenoptik Carl Zeiss JENA Gmbh
15	ISS 2000	Component tester	Schlumberger
16	S790	Board tester	Schlumberger
17	Z1800™ series	In-circuit board testers	Teradyne
18	Z8000™ series	Combinational board testers	Teradyne
19	L300™ series	Combinational and functional board testers	Teradyne

Tool Vendors with 1149.1 Support

	PRODUCT	DESCRIPTION	COMPANY
1	proTEST-PC™	PC-based boundary-scan test controller	Alpine Image Systems
2	Boundary Scan Master Evaluation Kit	PC-based hardware/ software kit used to evaluate the 497AA Boundary Scan Master and experiment with 1149.1	AT&T
3	TAPDANCE™	Generates test vectors for checking integrated circuit designs for conformance to 1149.1	AT&T
4	Test Assistant	Circuit synthesis tool which compiles 1149.1 logic	Compass Design Automation
5	SmartModel® Library	Behavioral models for board-level simulation	Logic Automation
6	Intelligen™	Circuit synthesis handling partial scan and ATPG	Racal-Redac
7	Test Compiler™	1149.1 architecture synthesizer and vector formatter	Synopsys
8	VICTORY™	Modular tool set for performing boundary-scan testing of PCBs, including automatic test generation and diagnostics	Teradyne
9	ASSET™ Diagnostic System	PC-based hardware/ software system for access and control of 1149.1 test structures for design verification and debug	Texas Instruments
10	ASSET™ Test System	PC hardware and run-time support software to support the application of ASSET test programs	Texas Instruments
11	ASSET™ Scan Function Library	A library of C++ routines used for accessing ASSET hardware from a custom C++ program	Texas Instruments
12	Scan Engine	Software component which, when coupled with an SN74ACT8990 forms the basis for an embedded 1149.1-driven self-test.	Texas Instruments
13	TDS®	Data transport environment supporting links to CAE and ATE for simulation and test	TSSI
14	BST-Explorer PM 3705	PC-based hardware/software for access and control of 1149.1 test structures.	FLUKE

Useful Literature and Tutorial Material

ORDER NUMBER	TITLE	DESCRIPTION	SOURCE
E1017-90001	HP Boundary-Scan Tutorial and BSDL Reference Guide	Includes BSDL syntax guide	Hewlett Packard
EH0321-0	The Test Access Port And Boundary-Scan Architecture	In depth description and application of 1149.1 with paper reprints	IEEE
SH13144	IEEE Standard Test Access Port and Boundary-Scan Architecture	IEEE Std 1149.1-1990	IEEE
	Serial Vector Format (SVF) Specification	Defines vector format for parallel and 1149.1 serial vectors for CAE and ATE exchange purposes	Texas Instruments
SATB002	Scan Educator	PC-based training software introducing 1149.1	Texas Instruments
SSYA002A	Testability Primer	1149.1 introduction and application notes	Texas Instruments
SSYA006	SCOPE™ Testability Products Applications Guide	Over 20 application notes and paper reprints	Texas Instruments

TRADEMARKS

proTest-PC is a trademark of Alpine Image Systems, Incorporated

TAPDANCE is a trademark of AT&T

GR-227x and GR-228x are trademarks of GenRad, Incorporated

HP 3070 and HP 3065AT are trademarks of Hewlett Packard Company

SmartModel is a registered trademark of Logic Automation Incorporated

SPARC is a trademark of Sun Microsystems, Incorporated

Test Compiler is a trademark of Synopsys Incorporated

VICTORY, L300, Z1800 and Z8000 are trademarks of Teradyne, Incorporated

TDS is a registered trademark of Test Systems Strategies, Incorporated

ASSET and SCOPE are trademarks of Texas Instruments Incorporated

Texas Instruments recognizes all trademarks used in this document.

APPENDIX B.

BIBLIOGRAPHY

- Andrews, J., "IEEE's P1149.5 Bus Facilitates JTAG," ATE and Instrumentation Conference, January 1992
- Andrews, W., "JTAG Works to Standardize Chip, Board and System Self-Test," Computer Design, July 1, 1989
- Archer, H.S., "A Comprehensive Analyzer for the JIAWG High Speed Data Bus," IEEE National Aerospace and Electronics Conference (NAECON), 1990
- Arment, E.L., Coombe, W.D., "Application of JTAG for Digital and Analog SMT," ATE and Instrumentation Conference West, 1989
- Aubert, J., "Boundary Scan Modification to Enhance Multichip Module Testing," National Aerospace Electronics Conference (NAECON), 1992
- Avra, L., "A VHSIC ETM-Bus-Compatible Test and Maintenance Interface," IEEE International Test Conference, 1987
- Ballew, W.D., Streb, L.M., "Board-Level Boundary Scan: Regaining Observability with an Additional IC," IEEE International Test Conference, 1989
- Bardier, P., d'Hervilly, G., "Performance Issues in HSDB Standard (SAE-AS-4074.1): How To Tune the HSDB Protocol," Real-Time Data Communications for Military Applications Conference, November 1990
- Barton, P. and Dolan, C., "ASICs and Testability Devices Revolutionize Testability Design," Texas Instruments Technical Journal, July/August 1988
- Beenker, F.P.M., "Systematic and Structured Methods for Digital Board Testing," IEEE International Test Conference, 1985
- Bennetts, R.G., Osseyran, A., "IEEE Standard 1149.1-1990 on Boundary Scan: History, Literature Survey, and Current Status," Journal of Electronic Testing: Theory and Applications, March 1991

- Bermingham, W.J., Fagotti, M.R., Rosen, W.A., Zeitz, R., "A Throughput and Latency Comparison of Linear and Ring Fiber Optics Buses," SPIE - The International Society for Optical Engineering, 1990
- Bhavsar, D.K., "Testing Interconnections to Static RAMs," IEEE Design & Test of Computers, June 1991
- Bicknell, J., "Advanced Avionic Communications for Military Use," IEE Colloquium on 'Time Critical Communications for Instrumentation and Control', November 1989
- Bicknell, J., "The High Speed Data Bus (for Military Avionics)," Military Avionics Architecture for Today and Tomorrow. 1988 Seminar Proceedings, November 1988
- Bingham, P., Asker, M., "ASIC Layout Software Builds in JTAG Rules," Electronics Manufacture & Test, March 1991
- Borland, A., "Experience of Designing JTAG ASICs Within System X," IEE Colloquium on 'Application and Development of the Boundary-Scan Standard', December 1990
- Breuer, M.A., Lien, J.C., "A Test and Maintenance Controller for a Module Containing Testable Chips," IEEE International Test Conference, 1988
- Brglez, F., Gloster, C., Kedem, G., "Hardware-Based Weighted Random Pattern Generation for Boundary Scan," IEEE International Test Conference, 1989
- Broderick, S., Wills, K., "The Intricacies of Boundary Scan Solutions," Evaluation Engineering, September 1991
- Brown, D.R., "The Development of an EFABus Ground Demonstration System," MIL-STD-1553B and the Next Generation, November 1989
- Brown, D.R., "Issues Concerning the Implementation of High Speed Optical Databus Systems," IEE Colloquium on 'Future Military Avionic Architectures', May 1990
- Brown, J., "Design of a Parallel Bus-to-Scan Test Port Converter," Electro International Conference, 1991
- Bruce, W.C., Gallup, M.G., Giles, G., Munns, T., "Implementing 1149.1 on CMOS Microprocessors," IEEE International Test Conference, 1991
- Caldwell, B., "Implementing 1149.1 Boundary Scan for Board Test," Test Engineering Conference, 1991

Caldwell, B. and Langford, T., "Is IEEE 1149.1 Boundary Scan Cost Effective: A Simple Case Study," IEEE International Test Conference, 1992

Collins, P., "Boundary Scan-the ATE Vendors' View," IEEE International Test Conference, 1988

Cortez, R., Dandapani, R., Yeager, M., "Issues of Integrating the IEEE Std 1149.1 into a Gate Array," VLSI Test Symposium, 1991

Covington, R.R., Goodwin, R.C., Reed, T.A., "Boundary-Scan Testing with ATEs," ATE and Instrumentation Conference, January 1992

Cron, A.D., "A Survey of Boundary-Scan Devices, Instrumentation, and Support Tools," NEPCON East, 1992

Cron, A.D., "IEEE-1149.1 Use in Design for Verification and Testability at Texas Instruments," The Second Annual IEEE ASIC Seminar and Exhibit, 1989

Crouch, A. and Pyron, C., "Impact of JTAG/P1149.1 Testability on Reliability," Government Microcircuit Applications Conference (GOMAC), 1989

Dahbura, A.T., Uyar, M.U., Yau, C.W., "An Optimal Test Sequence for the JTAG/IEEE P1149.1 Test Access Port Controller," IEEE International Test Conference, 1989

Daniel, W., Young, G., "VHSIC Testability: An IC- to System-Level Implementation," Texas Instruments Technical Journal, July-August 1988

Daniel, W., TI internal memorandum, October 1989

Daniel, W., Miller, E., "Implementing BIST and Boundary Scan on VHSIC/VLSI Designs Using the JTAG/IEEE P1149.1 Test Bus," Advanced Microelectronics Qualification, Reliability and Logistics Workshop, July-August 1989

Daniel, W., "Fault Emulation Using Boundary Scan," IEEE VLSI Test Symposium, April 1990

Daniel, W., "Design Verification of a High Density Computer Using IEEE 1149.1," IEEE International Test Conference, 1992

Daniel, W., "Embedding an IEEE 1149.1 Test Controller," Design and Test Expo, January 1993

- Davidson, S., "Merging BIST and Boundary Scan at the IC level," Wescon Conference, 1988
- De Jong, F., Matos, J.S., Ferreira, J.M., "Boundary Scan Test, Test Methodology, and Fault Modeling," Journal of Electronic Testing: Theory and Applications, March 1991
- De Jong, F., "BITL Format: A Way of Representing Boundary Scan Test Vectors," Electronics Letters, July 5, 1990
- De Jong, F., "Boundary Scan Test Used at Board Level: Moving Towards Reality," IEEE International Test Conference, 1990
- Dennis, P., "Advanced Diagnostic Architecture for JIAWG Compliant Designs," IEEE AUTOTESTCON, 1990
- Dervisoglu, B.I., "Using Scan Technology for Debug and Diagnostics in a Workstation Environment," IEEE International Test Conference, 1988
- Dervisoglu, B.I., "Boundary-Scan Update IEEE P1149.2 Description and Status Report," IEEE Design and Test of Computers, September 3, 1992
- Deshayes, J.G., "Boundary Scan Methods for Interconnection Testing on Electronic Modules," Elektronik, April 28, 1992
- Dettmer, R., "JTAG-Setting the Standard for Boundary-Scan Testing," IEEE Review, February 16, 1989
- DeSena, A., "Interface Brings Standardized Design for Testability Closer to Reality," Computer Design, May 15, 1988
- Dingle, S.L., Lacroix, L.D., Twombly, P.A., "The Advantages of Boundary-Scan Testing," VLSI Test Symposium, 1991
- Donnell, J., "Boundary Scan Puts Tomorrow's Devices to Test," Electronic Design, June 27, 1991
- Eichelberger, E. B. and Williams, T.W., "A Logic Design Structure for LSI Testability," Journal of Design Automation and Fault-Tolerant Computing, May 1978
- Ellis, M., Jr., Bell, B., "Bottom-Up Techniques Propel Board Testability," Electronic Design, May 24, 1990

Fasang, P.P., "Application of Boundary Scan to Analogue-Digital ASIC Test," Australian Electronics Engineering, June 1990

Fasang, P.P., "ASIC Testing in a Board/System Environment," IEEE Custom Integrated Circuits Conference, 1989

Fichtenbaum, M.L., Robinson, G.D., "Scan Test Architectures for Digital Board Testers," IEEE International Test Conference, 1990

Fitch, K.D., Kane, J., "Application of Boundary-Scan and Full-Chip BIST to a 3 ASIC Chip Set," IEEE Custom Integrated Circuits Conference, 1991

Fleming, P., "Expanding Beyond Boundary Scan Techniques and JTAG," Wescon Conference, 1988

Fleming, P., "Semiconductor Perspective on Test Standards," IEEE International Test Conference, 1988

Forrest Tones, C., "New Avionics ATE Considerations for the Boeing 777," ATE and Instrumentation Conference, January 1992

Gallup, M.G., Ledbetter, W., Jr., McGarity, R., McMahan, S., Scheuer, K.C., Shepard, C.G., Sood, L., "Testability Features of the 68040," IEEE International Test Conference, 1990

Gartner, P., Buchner, T., Roos, G., Schwederski, T., "Boundary Scan and its Application to the IMS Gate Forest," Journal of Semicustom ICs, December 1991

Grace, P., "Scan Design Gives Chip Level Diagnosis at System Test," New Electronics, March 1989

Graff, G., "Scanning the Boundaries of Test," Electronics Manufacture & Test, September 1989

Griffin, K., "VHSIC Phase 2 Test Requirements for the Depot," AUTOTESTCON, 1989

Grimberg, O., "An advanced Testability Concept for Space Applications," ESA Electronic Components Conference, 1990

Halliday, A., Young, G., Crouch, A., "Prototype Testing Simplified by Scannable Buffers and Latches," IEEE International Test Conference, 1989

- Hamlin, D.B., "The Unisys Implemented, JIAWG Compliant, Linear, Token-Passing HSDB Chip Set Description," IEEE National Aerospace and Electronics Conference (NAECON), 1991
- Hansen, P., "Implementing Boundary Scan Test Strategies," AUTOTESTCON, 1990
- Hansen, P., "Mixed Technology Boards Using Scan/Non-Scan Parts," Test, September 1990
- Hansen, P., "Testing Conventional Logic and Memory Clusters Using Boundary Scan Devices as Virtual ATE Channels," IEEE International Test Conference 1989
- Hansen, P., "Strategies for Testing VLSI Boards Using Boundary Scan," Electronic Engineering, November 1989
- Hansen, P., Borroz, T., "Tough Board Test Problems Solved With Boundary Scan," Electronics Test, June 1989
- Hansen, P., "Testing the Internal Silicon of Boundary-Scan Devices Using Boundary-Functional Test and Serial Vector Format," NEPCON West, 1992
- Hao, C.H., Scholz, H.N., Tulloss, R.E., Yau, C.W., Wach, W., "Computer Aided Structured Design for Testability of ASICs," 8th Australian Conference on Microelectronics, 1989
- Harrod, P.L., Biggs, J.P., "Boundary Scan Design for a Memory Controller," IEE Colloquium on 'Application and Development of the Boundary-Scan Standard', December 1990
- Hassan, A., Agarwal, V.K., Rajski, J., Dostie, B.N., "Testing of Glue Logic Interconnects Using Boundary Scan Architecture," IEEE International Test Conference, 1989
- Herrmann, J.J., "High Speed Data Bus Active Coupler," IEEE National Aerospace and Electronics Conference (NAECON), 1991
- Hewlett Packard, HP Boundary-Scan Tutorial and BSDL Reference Guide
- Hilla, S.C., "Boundary Scan Testing for Multichip Modules," IEEE International Test Conference, 1992

Hobbs, E.D., "Practical Considerations for Designing ASICs Which Incorporate Scan Path and JTAG Techniques," IEE Colloquium on 'Automated Testing and Software Solutions', April 1992

Hobson, R.F., "Combining Boundary Scan With I/O and Other System Functions to Reduce System Complexity," Microelectronics Journal, May 1992

Hughes, J.L.A., Pahlajrai, P., "Effects of Packaging and Interconnect Technology on Testability of Printed Wiring Boards," SPIE - The International Society for Optical Engineering, 1991

IBM, Honeywell, and TRW, VHSIC Phase 2 Interoperability Standards: TM-Bus Specification -- Version 3.0, 1987

IEEE/ANSI 896 Standard, Futurebus+

IEEE/ANSI 960 Standard, FASTBUS

IEEE/ANSI 1014 Standard, VMEbus

IEEE/ANSI 1295 Standard, Multibus II

IEEE Proposed Standard P1394, High Speed Serial Bus

IEEE Proposed Standard P1149.2 Draft: IEEE Extended Digital Serial Subset

IEEE Proposed Standard P1149.4: IEEE P1149.4 Mixed-Signal Test Bus Standards

IEEE Proposed Standard P1149.5 Draft: IEEE Standard Module Test and Maintenance (MTM) Bus Protocol

IEEE Standard 1149.1-1990: IEEE Standard Test Access Port and Boundary Scan Architecture, 1990

Jackson, C., "Fiber Optics Gives Avionics a Lift," Photonics Spectra, August 1989

Jacob, G., "Functional Board ATE," Evaluation Engineering, March 1990

Jarwala, N., "Boundary Scan Promises Test Alternatives," Design Automation, April 1991

- Jarwala, N., Yau, C.W., "A New Framework for Analyzing Test Generation and Diagnosis Algorithms for Wiring Interconnects," IEEE International Test Conference, 1989
- Jarwala, N. and Yau, C.W., "Achieving Board-Level BIST Using the Boundary-Scan Master," IEEE International Test Conference, 1991
- Jones, T.D., Millward, R., "An Initial Design Study of the Use of Boundary Scan to Simplify System Test (guided weapons)," IEE Colloquium on 'Application and Development of the Boundary-Scan Standard', December 1990
- Jung-Cheun Lien, Breuer, M.A., "An Optimal Scheduling Algorithm for Testing Interconnect Using Boundary Scan," Journal of Electronic Testing: Theory and Applications, March 1991
- Kajitani, H., Sato, H., Saito, H., Oresjo, S., "Practical Test Generation With IEEE 1149.1 Boundary-Scan," ATE and Instrumentation Conference, January 1992
- Kar, S., Chu, M., "Software Approaches to Test Scan-Related Boards," ATE and Instrumentation Conference West, 1989
- Karpenske, D., Talbot, C., "Testing and Diagnosis of Multichip Modules," Solid State Technology, June 1991
- Koeter, J., "Designing IEEE 1149.1 Compatible Boundary-Scan Logic Into an ASIC Using Texas Instrument's Scope Architecture," Third Annual IEEE ASIC Seminar and Exhibit, 1990
- Ladner, D.C., "Reconfigurable Hardware-From Design to Manufacturing Test," Wescon Conference, 1988
- Landis, D.L., Singh, P., "A Wafer Scale IEEE 1149.1 Case Study," Test Engineering Conference, 1991
- Landis, D.L., "A Self-Test Methodology for Restructurable WSI," International Conference on Wafer Scale Integration, 1990
- Landis, P., "Applications of the IEEE P1149.5 Module Test and Maintenance Bus," IEEE International Test Conference, September 1992
- Lee, J., "Desk-Top Boundary Scan Test," ATE and Instrumentation Conference, January 1992

Lefebvre, M.E., "Functional Test and Diagnosis: A Proposed JTAG Sample Mode Scan Tester," IEEE International Test Conference, 1990

Lester, R., Wasserman, S., "Implementing JTAG Boundary Scan With Methodologies Which Minimize Design Overhead," Wescon Conference, 1989

Levy, A., "Industrial Testability Standard Fits Military Applications," Electronic Products, October 1989

Ludemann, U., Vogt, H., "Boundary Scan-A User's Point of View," Wescon Conference, 1988

Ludvigson, M.T., "Thoughts on High Speed Data Bus Performance," IEEE National Aerospace and Electronics Conference (NAECON), 1990

Maierhofer, J., "Hierarchical Self-Test Concept Based on the JTAG Standard," IEEE International Test Conference, 1990

Mansoorian, B., Shooktim, R., Lee, L.S., Shahrokhinia, S., "BiCMOS Testability Circuits for IEEE 1149.1," Bipolar Circuits and Technology Meeting, 1991

Marshall, J., "A PC-Based JTAG Test Environment," Electro International Conference, 1991

Matos, J., Pinto, F. and Ferreira, J., "A Boundary Scan Test Controller for Hierarchical BIST," IEEE International Test Conference, 1992

Maunder, C., "A D&T Special Report-Boundary Scan: And End-of-Term Report-IEEE Std 1149.1 Survey Results," IEEE Design & Test of Computers, June 1992

Maunder, C.M., Tulloss, R.E., "Testability on TAP," IEEE Spectrum, February 1992

Maunder, C., "Applications and Onward Development of ANSI/IEEE Std 1149.1," IEE Colloquium on 'Application and Development of the Boundary-Scan Standard', December 1990

Maunder, C and Tulloss, R., The Test Access Port and Boundary Scan Architecture, IEEE Computer Society Press, 1990

McBean, D., Moore, W., "Bridging Fault Algorithms for a Boundary Scan Board," IEE Colloquium on 'Application and Development of the Boundary-Scan Standard', December 1990

McClean, D., Romeu, J., "Design for Testability With JTAG Test Methods," Electronic Design, June 8, 1989

Mejzak, R.S., "JIAWG Diagnostic Concept and Commonality Requirements," IEEE National Aerospace and Electronics Conference (NAECON), 1990

Meyer, J., "Society of Automotive Engineers AS4074 Family High-Speed, Fault-Tolerant Data Communications Standards for Integrated Avionics," 9th Digital Avionics Systems Conference, October 1990

Miller, B., "Scan Conversion of ASICs," Circuit Design, February 1990

Miske, M., "Addressing Basic 1149.1 Design Concerns," NEPCON, 1992

Mohaswaran, M., "Scan Design Gives Time to Market Edge," New Electronics, July-August 1990

Moore, T.M., "A Workstation Environment for Boundary-Scan Interconnect Testing," IEEE International Test Conference, 1991

Modrow, M., Hatfield, D., "High Speed Bus Technology Development," Wright Research and Development Center, Contract No.: F33615-83-C-1036; 2734; 02, September 1989

Morgan, R.J., "System Testability Using Standard Logic," Electro Conference, 1990

Morley, S., Scharf, J., "Software Support for Boundary-Scan Techniques," Electronics Test, June 1990

Moxon, T.W., "Design Trade-Offs When Implementing Boundary Scan in an Application Specific Integrated Circuit," Third Annual IEEE ASIC Seminar and Exhibit, 1990

Muris, M., "Integrating Boundary Scan Test Into an ASIC Design Flow," IEEE International Test Conference, 1990

Nagvajara, P., Karpovsky, M.G., Levitin, L.B., "Pseudorandom Testing for Boundary-Scan Design With Built-In Self-Test," IEEE Design & Test of Computers, September 1991

Nelson, J.H., Shafer, L.T., Hamlin, D.B., "Performance Analysis for a Candidate Linear Token-Passing, High-Speed Data Bus," IEEE National Aerospace and Electronics Conference (NAECON), 1988

Nightingale, D., "Implementing Boundary Scan Testing With ASICs," Electronics Manufacture & Test, March 1990

Oakland, S.F., "Combining IEEE Standard 1149.1 With Reduced-Pin-Count Component Test," VLSI Test Symposium, 1991

Oresjo, S., "The Boundary Scan Alternative to Testing Surface Mount PC boards," Surface Mount Exposition and Conference, 1990

Oresjo, S., "Results of Using the 1149.1 Boundary-Scan Standard," NEPCON West, 1992

Parker, K.P., Oresjo, S., "A Language for Describing Boundary-Scan Devices," IEEE International Test Conference, 1990

Parker, K.P., "Production Board Testing in a Boundary Scan Environment," Australian Electronics Engineering, September 1990

Patel, K., "Data Bus Testing Through the Stages," MIL-STD-1553B and the Next Generation, November 1989

Pearce, T.H., "Signal Processor Data Bus Requirements for Integrated Communications Architectures (Military Aircraft)," Military Avionics Architecture for Today and Tomorrow Seminar, 1988

Posse, K.E., "A Design-for-Testability Architecture for Multichip Modules," IEEE International Test Conference, 1991

Prohofsky, T., "High Speed Data Bus Macro Instruction Set Architecture," IEEE National Aerospace and Electronics Conference (NAECON), 1991

Pyron, C., Sallade, R., "Diagnostic Verification," IEEE International Test Conference, September 1989

Quinnell, R.A., "JTAG Boundary-Scan Test: Adding Testability Also Aids Debugging," Electronic Design News (EDN), August 2, 1990

Raymond, D., "Boundary Scan and ASIC Vectors in a Low Cost In-Circuit System," Electronics Manufacture & Test, Dec. 1990-Jan. 1991

Reed, C., "ATE Fixturing Update: Fixturing in the 1990s," Evaluation Engineering, September 1990

Rich, B.A., Bartels, B.E., Cole, M.H., "Maintenance Technology for Advanced Avionics Architecture," IEEE National Aerospace and Electronics Conference (NAECON), 1990

Richard, D., Anderson, G., McIver, G., "Very High Speed Integrated Circuits (VHSIC). Phase 2. Submicrometer Technology Development, Interoperability Standards," Contract No.: DAAK20-85-C-0376; 2700, November 30, 1988

Richards, D.J., "Value of Testability Standards in Testing Commercial Products," IEEE International Test Conference, 1988

Robinson, G., "Decision to Use Scan Must be a Joint One," Electronics Manufacture & Test, September 1990

Robinson, G.D., Deshayes, J.G., "Interconnect Testing of Boards With Partial Boundary Scan," IEEE International Test Conference, 1990

Robinson, J.P., "Circular Built-In Self-Test," Northcon Conference, 1989

Robinson, J.P., "Test Design and Boundary Scan," Southcon Conference, 1989

Rogel-Favila, B., "Automatic Test Generation and Fault Diagnosis of Boundary Scan Circuits," IEE Colloquium on 'Automated Testing and Software Solutions', April 1992

Runyon, S., "Boundary-Scan-Tool Users Are Doing It Themselves," Electronic Engineering Times, February 11, 1991

Russell, R.J., "The JTAG Proposal and Its Impact on Automatic Test," ATE and Instrumentation Conference East, 1988

Sallade, R., et al, "Built In Test - Requirements, Issues and Architectures," Texas Instruments Technical Journal, July-August 1989

Sallade, R., "Lessons Learned in Applying IEEE 1149.1 Testability in Defense System Programs at Texas Instruments," Test Engineering Conference, 1991

Samad, M.A., "A Toolbox For ASIC Testability Automation," IEEE Custom Integrated Circuits Conference, 1990

Scheiber, S.F., "Test Tactics for Partial-Scan Boards," Test & Measurement World, April 1991

Scholz, H.N., Tulloss, R.E., Yau, C.W., Wach, W., "ASIC Implementations of Boundary-Scan and Built-In Self-Test (BIST)," Journal of Semi-Custom ICs, June 1989

Sedmak, Richard M., "Practical Considerations in BIST and Boundary-Scan Implementation," Design and Test Expo, January 1992

Setty, A.A., Martin, H.L., "BIST and Interconnect Testing With Boundary Scan," IEEE SOUTHEASTCON, 1991

Sharma, R., "Test Generation For Structural Testing With Boundary Scan," ATE and Instrumentation Conference, January 1992

Sherratt, C.J., "ICL's First Development Using IEEE 1149.1 (JTAG)," IEE Colloquium on 'Application and Development of the Boundary-Scan Standard', December 1990

Sicola, S., "JTAG (IEEE 1149.1) and a Self-Testing System," Test Engineering Conference, 1991

Siguenza, A.V.I., "A Self-Test and Boundary-Scan Controller Chip Using the ETM Bus Specifications," Northcon Conference, 1989

Smith, G.J., "Component Test Interface Survey," ATE and Instrumentation Conference West, 1989

Spohrer, T., Marquette, D., Gallup, M., "Test Architecture of the Motorola 68040," Proceedings. IEEE International Conference on Computer Design: VLSI in Computers and Processors, 1990

Sterba, D., Halliday, A., McClean, D., "ATPG and Diagnostics For Boards Implementing Boundary Scan," Journal of Electronic Testing: Theory and Applications, March 1991

Stevens, R., "High Speed Data Bus Design Validation," National Aerospace and Electronics Conference (NAECON), 1991

Teradyne and Texas Instruments, Inc., Serial Vector Format (SVF) Specification

Texas Instruments, "SCOPE Testability Products: Applications Guide," 1990

Thompson, P., "Coping With the Move to Boundary Scan Test," Electronics Manufacture & Test, April 1990

Trotter, P., LaPadula, L., Pawlowski, G. and Rekieta, D., "Boundary Scan and BIST in Large High Speed ASIC," Government Microcircuit Applications Conference (GOMAC), 1989

Tulloss, R.E., "Market Forces Driving Acceptance of ANSI/IEEE Std 1149.1-1990 Boundary-Scan," Electro International Conference, 1991

Tulloss, R.E., Yau, C.W., "BIST and Boundary-Scan For Board Level Test: Test Program Pseudocode," 1st European Test Conference, 1989

Turino, J., "Design For Test Using Boundary Scan and Testability Buses," Surface Mount Technology, April 1991

Uhlhorn, R.W., "The Fiber-Optic High-Speed Data Bus For a New Generation of Military Aircraft," IEEE LCS, February 1991

Uhlhorn, R.W., McDermott, T.A., Goldman, P.C., "An Overview of the Fiber-Optic Active Star Coupler program," IEEE National Aerospace and Electronics Conference (NAECON), 1990

Uhlhorn, R.W., "A Robust Fiber Optic Active Star Coupler For the SAE Linear Token-Passing Multiplex Data Bus," IEEE Aerospace and Electronics Systems Magazine, January 1989

Uhlhorn, R.W., "Fiber Optic Buses and Networks For Advanced Avionics Architectures," Computing Systems Configuration for Highly Integrated Guidance and Control Systems, 1988

Van de Goor, A.J., van Tetering, J.A.M., "A Low-Cost Tester For Boundary Scan," Microprocessors and Microsystems, March 1991

Van de Lagemaat, D., "Testing Multiple Power Connections With Boundary Scan," 1st European Test Conference, 1989

Van Riessen, R.P., Kerkhoff, H.G., Kloppenburg, A., "Design and Implementation of a Hierarchical Testable Architecture Using the Boundary Scan Standard," 1st European Test Conference, 1989

Vining, S., "Tradeoff Decisions Made For a P1149.1 Controller Design," IEEE International Test Conference, 1989

Wagner, P.T., "Interconnect Testing With Boundary-Scan," IEEE International Test Conference, 1987

Wang, L.-T., Marhoefer, M., McCluskey, E.J., "A Self-Test and Self-Diagnosis Architecture For Boards Using Boundary Scans," 1st European Test Conference, 1989

Whetsel, L., "At-Speed Board Test Simplified via Embeddable Data Trace/Compaction IC," IEEE AUTOTESTCON, September 1991

Whetsel, L., "A Proposed Method of Accessing 1149.1 in a Backplane Environment," IEEE International Test Conference, September 1992

Whetsel, L., "An IEEE 1149.1 Backplane Access Approach," Design and Test Conference, January 1993

Whetsel, L., "Proposed Updates to the IEEE 1149.1 Standard," ATE and Instrumentation Conference, 1992

Whetsel, L., "JTAG Compatible Devices Simplify Board Level Design For Testability," Wescon Conference, 1989

Whetsel, L., "A Proposed Standard Test Bus and Boundary Scan Architecture," Wescon Conference, 1988

Whetsel, L., "Event Qualification: A Gateway to At-Speed Functional Testing," IEEE International Test Conference, 1990

Whetsel, L., "An IEEE 1149.1 Based Logic/Signature Analyzer in a Chip," IEEE International Test Conference, 1991

Wilkins, B.R., "The IED Boundary Scan Project," IEE Colloquium on 'Application and Development of the Boundary-Scan Standard', December 1990

Woppman, G., "Economics of Boundary Scan," Computer Design News, 1990

Yau, C.W., Jarwala, N., "The Boundary-Scan Master: Target Applications and Functional Requirements," IEEE International Test Conference, 1990

Yau, C.W., Jarwala, N., "A Unified Theory For Designing Optimal Test Generation and Diagnosis Algorithms For Board Interconnects," IEEE International Test Conference, 1989

Zorian, Y., Chi Yau, "Linking BIST and Boundary Scan to Test Success," Test & Measurement World, May 1991

Zorian, Y., Jarwala, N., "Designing Fault-Tolerant, Testable, VLSI Processors Using the IEEE P1149.1 Boundary-Scan Architecture," IEEE International Conference on Computer Design, 1989

APPENDIX C.

ACRONYMS

3SB.....	3-State Buffer
A/D.....	Analog to Digital
AAD.....	Assign ADdress
AATD.....	Aviation Applied Technology Directorate
AB1.....	Analog Bus 1
AB2.....	Analog Bus 2
AC.....	Alternating Current
AH.....	Acceptor Handshake
AI.....	Address Input bus
ANSI.....	American National Standards Institute
AO.....	Address Output bus
APO.....	Acknowledge Protocol Output signal
APS.....	F-22 Radar Array Power Supply
APSC.....	Array Power Supply Control
AS.....	Analog Switch
ASCII.....	American Standard Code for Information Interchange
ASI.....	Analog Scan In
ASIC.....	Application Specific Integrated Circuit
ASO.....	Analog Scan Out
ASP.....	Addressable Shadow Port
ASSET(tm).....	Advanced Support System for Emulation and Test
AT&T.....	American Telephone & Telegraph
ATDI.....	Analog Test Data In
ATDO.....	Analog Test Data Out
ATE.....	Automatic Test Equipment
ATN.....	ATteNtion
BC/RT.....	Bus Controller/Remote Terminal
BiCMOS.....	Bipolar Complementary Metal Oxide Semiconductor

BIST..... Built-In Self-Test
 BIT..... Built-In Test
 BM..... Bus Management
 BNF..... Backus-Naur Form
 BPM..... Basic Processing Module
 BR..... Bypass Register
 BSC..... Boundary-Scan Cells
 BSDL..... Boundary Scan Description Language
 BYPASS..... IEEE 1149.1 BYPASS instruction
 C..... Controller
 CAB..... Common Avionics Baseline
 CAE..... Computer Automated Engineering
 CBIT..... Continuous Self-Test Built-in Test
 CLK..... system CLock
 CLS..... CLear Status
 CMPOUT..... CoMPare OUTput
 CND..... CanNot Duplicate
 Commanche..... Light Helicopter
 CPU..... Central Processing Unit
 CRC..... Cyclic Redundancy Code
 CUT..... Circuit Under Test
 CWG..... Commonalty Working Group
 D/A..... Digital to Analog
 DAB..... DAta Byte
 DAC..... Data ACcepted
 DAC..... Digital to Analog Converter
 DARPA..... Defense Advanced Research Projects Agency
 DAV..... DAta Valid
 dB..... deciBels
 DC..... Direct Current
 DC or DCL..... Device CLear
 DDT..... Define Device Trigger

DFT Design For Testability
 DIB Device Identification Bits
 DIO Discrete I/O
 DIP Dual Inline Package
 DLF Disable Listener Function
 DMA Direct Memory Access
 DMC Define MaCro
 DoD Department of Defense
 DR Data Register
 DSP Digital Signal Processors
 DT Device Trigger
 EEPROM Electrically Erasable Programmable Read Only Memory
 EL Extended Listener
 EMC (IEEE P1149.5) MTM-bus Enable Module Control command
 EMI Electromagnetic Magnetic Interference
 END END
 EOI End Or Identify
 EPROM Erasable Programmable Read Only Memory
 EQC Event Qualification Cell
 EQI Event Qualification Input
 EQM Event Qualification Module
 EQO Event Qualification Output
 EQUAL Event QUALification architecture
 ESE Event Status Enable
 ETM-Bus Element Test and Maintenance Bus
 EVT EVenT
 EXT EXTension bit
 EXTEST IEEE 1149.1 EXternal TEST instruction
 F22 Advanced Tactical Fighter
 FET Field Effect Transistor
 FF Flip-Flop
 FM Failures/Million hours

FPGA..... Field Programmable Gate Array
 GET..... Group Execute Trigger
 GFLOPS Giga FLoating point OPerations per Second
 GMU Gigabit Memory Unit
 GPIB..... General Purpose Interface Bus
 GTL..... Go To Local
 HDL..... Hardware Description Language
 HP Hewlett Packard
 HSDB High Speed Data Bus
 HSDL Hierarchical Scan Description Language
 Ht Hierarchical Testability
 I/O Input/Output
 IBIT..... Initiated Built-in Test
 IBM International Business Machines
 IC..... Integrated Circuits
 ICE In-Circuit-Emulation
 ICT In Circuit Test
 ID IDentification
 IDCODE..... IEEE 1149.1 IDentification CODE instruction
 IDN..... IDeNtification query
 IDR..... Implementation Detail Register
 IDY..... IDentifY
 IEEE..... Institute of Electrical and Electronics Engineers
 IEEE Std 1149.1 Standard Test Access Port and Boundary Scan Architecture
 IEEE Std 488..... see GPIB
 IEEE Std P1149.2..... Extended Digital Serial Test Bus
 IEEE Std P1149.4..... Mixed-Signal Test Bus
 IEEE Std P1149.5..... Standard Module Test and Maintenance Bus Protocol
 IEEE Std P1394..... High Speed Serial Bus
 IFA Input Filter Assembly
 IFC InterFace Clear
 IMP..... International Microelectronics Products

in..... inch
 INTEST IEEE 1149.1 Internal TEST instruction
 IOBD Integrated On-Board Diagnostics
 IR..... Instruction Register
 ISA Instruction Set Architecture
 ISO ISOchronous
 JIAWG..... Joint Integrated Avionics Working Group
 JTAG Joint Test Action Group
 KBPS..... Kilo Bits Per Second
 kHz kiloHertz
 L Listener
 L/S..... Load/Scan control signal
 LFSR Linear Feedback Shift Register
 l-l line-to-line
 LL..... Link Layer
 LLO Local LockOut
 LRM Line Replaceable Module
 LSB Least Significant Bit
 LSSD Level Sensitive Scan Design
 mA..... milliAmpere
 MBPS Million Bits Per Second
 MBS Memory Bank Substrate
 MCLK (IEEE P1149.5) MTM-bus CLoCK
 MCM Multi-Chip Module
 MCTL..... (IEEE P1149.5) MTM-bus ConTroL
 MDU Memory Driver Unit
 MFLOPS..... Mega FLoating point OPERations per Second
 MHz..... MegaHertz
 MICT..... Module I/O Control and Test
 MIL-HDBK-217E..... Military HanDBooK 217E
 MIL-Std-1553B Multiplexed Data Bus
 MIL-Std-1773..... Fiber Optics Command/Response Multiplex Data Bus

MIPS Million Instructions Per Second
MLA My Listen Address
MMC (F-16) Modular Mission Computer
MMD (IEEE P1149.5) MTM-bus Master Data
MPR Module Pause Request
ms milliseconds
MSA My Secondary Address
MSB Most Significant Bit
MSD (IEEE P1149.5) MTM-bus Slave Data
MTA My Talk Address
MTBF Mean Time Between Failure
MTM-bus Module Test and Maintenance bus
MUX MultipleXer
MX MultipleXer
NDAC Not Data ACcepted
NOP No OPeration
NRFD Not Ready For Data
ns nanosecond
OPC OPeration Complete
OPT OPTion ID query
OSA Other Secondary Address
OTA Other Talk Address
p-p peak to peak
PBIT Periodic Built-in Test
PC Personal Computer
PC-AT Personal Computer-Advanced Technology
PCB Pass Control Back
PCB Printed Circuit Board
PCG Primary Command Group
PI Primary Input
PI-bus Parallel Intermodule bus
PIC Processor Interface Chip

PICC..... Processor Interface Control and Communications
 PIPO..... Parallel-In/Parallel Out
 PISO..... Parallel Input/Serial Output
 PO Primary Output
 PP Parallel Poll
 PPC..... Parallel Poll Configure
 PPD Parallel Poll Disable
 PPE..... Parallel Poll Enable
 PPR..... Parallel Poll Response
 PPU Parallel Poll Unconfigure
 PRE Parallel Poll Enable Register
 PRPG..... Pseudo Random Pattern Generator
 PRST Power up ReSeT circuit
 PSA Parallel Signature Analysis
 PSA Physical Station Address
 PSAR..... Parallel Signature Analysis Register
 PSC..... Power on Status Clear
 PTCK..... Primary port TCK
 PTDI..... Primary port TDI
 PTDO Primary port TDO
 PTMS Primary port TMS
 PTU PI-Bus/TM-Bus Unit
 PUD..... Protected User Data
 PWB Printed Wiring Board
 R/T Run/Test control signal
 RAM..... Random Access Memory
 RCL..... ReCaLL instrument state
 RCR..... ReCeiveR circuit
 RDT..... Resource Description Transfer
 REN..... Remote ENable
 RFD Ready For Data
 RISC..... Reduced Instruction Set Computer

RL..... Remote Local
 ROM..... Read Only Memory
 RQS ReQuest Service
 RST ReSeT
 RSTA..... ReSeT Address
 RT/IDLE..... Run Test/IDLE
 RTOK..... Re-Test OK
 RUNBIST IEEE 1149.1 RUN BIST instruction
 SAE Society of Automotive Engineers
 SAMPLE/PRELOAD IEEE 1149.1 SAMPLE or PRELOAD instruction
 SAP Scan Access Port
 SRAM..... Static Random Access Memory
 SAV SAVe instrument state
 SBIT Start-up Built-in Test
 SC..... SemiConductor
 SCI Serial Bus Interface
 SDC..... Selected Device Clear
 SEM-E..... Standard Electronic Module-size E
 SH Source Handshake
 SI..... Serial Input
 SIPO Serial Input/Parallel Output
 SM-Bus..... System Maintenance Bus
 SMT Surface Mount Technology
 SO Serial Output
 SP-bus..... Signal Processing bus
 SPD Serial Poll Disable
 SPE..... Serial Poll Enable
 SPS Scan Path Selector
 SR or SRQ..... Service ReQuest
 SRE Service Request Enable
 SSA Serial Signature Analysis
 SSR..... Solid State Recorder

STB SStatus Byte
 STCK..... Secondary port TCK
 STDI..... Secondary port TDI
 STDO Secondary port TDO
 STM Select Test Mode
 STMS Secondary port TMS
 SVF Serial Vector Format
 T..... Talker
 TAB..... Tape Automated Bonding
 TAP Test Access Port
 TBC..... Test Bus Controller
 TBE..... Test Bus Evaluation
 TCK..... IEEE 1149.1 Test Clock
 TCT Take ConTrol
 TDI..... IEEE 1149.1 Test Data Input
 TDO IEEE 1149.1 Test Data Output
 TE..... Extended Talker
 TI..... Texas Instruments
 TL..... Transaction Layer
 TLRST..... Test Logic ReSeT state
 TM-Bus (module level) Test and Maintenance Bus
 TMS IEEE 1149.1 Test Mode Select
 TRG..... TRiGger
 TRST IEEE 1149.1 Test ReSeT
 TSMD..... Time Stress Measurement Device
 TST Self-TeST query
 TTL Transistor-Transistor Logic
 UNL..... UNListen
 us microseconds
 USERCODE IEEE 1149.1 USER CODE instruction
 UUT Unit Under Test
 V..... Voltage

VCP..... Vector Co-Processor
Vdc..... Voltage of direct current
VHDL..... VHSIC Hardware Description Language in IEEE Std 1076-1987
VHSIC..... Very High Speed Integrated Circuit
VLSI..... Very Large Scale Integration
VMS (F-22) Vehicle Management System
WAI..... WAIt to complete
WSI Wafer Scale Integration
XBAR..... CrossBAR switch
XMT..... TransMiTter circuit